ABSTRACT

Title of dissertation: PERCEPTUAL GRAPHICS
FOR EFFECTIVE VISUALIZATION

Chang Ha Lee, Doctor of Philosophy, 2005

Dissertation directed by: Professor Amitabh Varshney
Department of Computer Science

Current trends in 3D graphics point to a near future when our ability to generate 3D content will far surpass our ability to analyze it meaningfully. These trends have inspired us to improve the comprehensibility of 3D graphics rendering using insights from human perception of geometry and illumination. In this dissertation, we develop algorithms and systems to seamlessly integrate the low-level human visual system cues with object modeling and lighting for 3D graphics.

Artists and illustrators have enhanced the perception of shape with discrepant lighting for centuries. Traditional graphics however assumes a model of consistent lighting. We have developed a lighting design system, that by relaxing the constraint of consistent lighting is able to convey a better depiction of shape. Our system for automatic lighting design, Light Collages, segments the objects into local surface patches and uses careful placement of highlights, shadows, and silhouettes on them to enhance shape perception. We have developed a spherical-harmonics-based formulation to achieve a 20-fold improvement in speed.

Geometric processing in graphics has made significant advances over the last

decade in defining and using mathematical measures of shape, such as curvature. However, less attention has been paid to the use of perception-inspired metrics for geometric processing. We have brought perception-inspired metrics to bear on the problem of processing and viewing for 3D graphics. We have developed the concept of scale-dependent mesh saliency for graphics. We have also explored how saliency can be used to prioritize operations in applications such as object simplification and to automatically compute desirable viewing parameters for 3D graphics applications.

We believe that Perceptual Graphics could lead us in the direction of more effective graphics applications that not only use computational resources wisely, but also lift the burden of unnecessary rendered detail from the human visual system.

PERCEPTUAL GRAPHICS FOR EFFECTIVE VISUALIZATION

by

Chang Ha Lee

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2005

Advisory Committee:

Professor Amitabh Varshney, Chair/Advisor
Professor David Jacobs
Professor Joseph JaJa
Professor David Mount
Professor Sergei Sukharev

ACKNOWLEDGMENTS

I would like to thank all the people who have helped me in one way or another to this dissertation. First I am deeply grateful to my advisor Dr. Amitabh Varshney. His passion and insights inspired me to enter the exciting area of computer graphics, and his excellent advice and warm encouragement were crucial to complete this work. In addition, he is a great mentor providing lifetime advice and lessons for me to be a better human being. I would also like to thank Dr. David Jacobs. He has provided invaluable guidance and support during the final stages of my graduate studies and his deep insights on human perception and computer vision were a great help to develop new ideas.

I would like to thank Dr. Joseph JaJa, Dr. David Mount, and Dr. Sergei Sukharev for serving on my committee. Joseph has provided inspiring questions and comments. David taught me fundamental concepts of computer science including algorithms and computational geometry through his classes. I also appreciate his wonderful comments and suggestions on research topics as well as his support on my career development. Sergei has provided interesting test models in computational biology and his different point of view on the dissertation topic was a great help and support. I would also like to thank Dr. Tapas Kanungo not only for giving me opportunities to work on interesting projects at the first stage of my graduate studies but also for his help and support even after he moved to IBM.

My colleagues at the graphics and visual informatics laboratory have enriched my graduate life in many ways. I owe my gratitude to Thomas Baby, Xuejun Hao, Derek Juba, Aravind Kalaiah, Youngmin Kim, and Zhiyun Li for their help and friendship. I also appreciate all other friends in the University of Maryland for sharing my non-research life.

I owe my deepest gratitude to my family. My parents and sisters have always stood by me and guided me through my life. Words cannot express the gratitude I owe them. Finally, my thanks and love go to my wife and daughter for their love and support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 Motivation

The last decade has seen a phenomenal growth in the complexity of 3D graphics models. An important reason behind this has been the advances in model acquisition technologies such as laser scanning, computer-vision-based reconstruction techniques, and high-precision medical imaging technologies. We are currently witnessing early stages in the development of large-scale wide-area sensor networks which are likely to increase 3D geometric content by several orders of magnitude. The accelerating rate of growth of 3D models in number and complexity creates a need for more sophisticated tools and techniques for storing, accessing, analyzing, visualizing, and interpreting such data effectively. Many interactive graphics applications have limited resources for graphics rendering. For example, we have a limited budget for the number of triangles per frame for interactive frame rates, limited resolution for the display devices, and limited visual comprehension abilities of the human visual system. Effective visualization can help us analyze and comprehend the visual information by omitting the irrelevant and emphasizing the important [2]. An important issue here is how we can classify what to omit and what to emphasize. Since the human perception is the last stage of the graphics rendering, incorporating lessons from research on human perception into graphics

processing should be helpful for improving computational and perceptual efficiency of graphics and visualization.

1.2 Dissertation Hypothesis

The hypothesis of this dissertation is as follows:

> *Visual emphasis techniques inspired by low-level human visual cues can facilitate generation of effective and compelling 3D graphics renderings.*

To examine the validity of this hypothesis, we have developed techniques for enhancing the visualization and geometric processing of 3D objects using insights from human perception of geometry and illumination. We have focused on effective processing and visualization in two ways.

First, we reduce the amount of visual information presented to the viewer. Recent research by Ostrovsky *et al.* [67] shows that human visual system perceives the illumination cues locally. Our automatic lighting design system elucidates geometric details by using globally discrepant lighting. We also introduce salient lighting to highlight important regions and suppress unimportant ones.

Second, we present a perception-inspired metric for assessing regional visual importance for 3D objects. The low-level human visual attention focuses retinal resources on interesting regions for efficiency. Many tasks in graphics and visualization can take advantage of similar principles to improve computational and visual efficiency. We have used this idea to improve several graphics applications such as

simplification, viewpoint selection, and lighting design, by identifying interesting regions.

1.3 Geometry-Dependent Lighting

1.3.1 Problem Definition

Our ability to generate 3D data, through acquisition and through simulation, has far surpassed our ability to visually comprehend it. As the data sizes continue to increase at a geometric rate of growth, it becomes necessary for us to revisit the traditional visualization pipeline to explore its stages that we can modify to enhance the comprehension of intricate model details. We believe that careful lighting design offers one such avenue of research.

Lighting design has long been considered crucial in conveying the right ambience, emotion, visual complexity, context, and in guiding the viewer's attention in art, scientific illustration, photography, stage lighting, and cinematography. Over two millennia ago Pliny the Elder discussed locally shading a surface fold to make it appear to rise above the background [27]. Since then, artists and illustrators have successfully used local lighting techniques for conveying the object shape. These local techniques convey a powerful impression of geometry, although the lighting across the surface is inconsistent.

Since the world around us is lighted consistently, it was possible that we might have naturally acquired the ability to discern illumination inconsistencies of lighting directions. However, recent research by Ostrovsky *et al.* [67] found that human sub-

jects were largely insensitive to illumination inconsistencies across a set of randomly-oriented 3D cubes. This helps explain why the geometry of consistent lighting is not as meticulously crafted in art as the geometry of perspective. There are also other reasons why artists and illustrators may allow lighting to be inconsistent. First, efforts to ensure consistent lighting in art are usually under-appreciated since they are not visually obvious. Second, artists can use inconsistent lighting to guide the viewer's attention to enhance comprehensibility or convey their message. If one were to apply the inverse lighting models that have been developed recently [7, 75, 102] to most paintings and illustrations, one would find innumerable errors (some admittedly slight, but present nonetheless) in their lighting and shading. However, not only have these lighting errors passed virtually unnoticed by most untrained human observers, lighting for such paintings is visually impressive and sometimes even deeply compelling.

Cavanagh [12] has suggested that our brain perceives the shape-from-shading cues locally and does not use large regions of the visual field for shape-from-shading analysis. In fact, recent work by Akers *et al.* [3] and Agarwala *et al.* [1] has shown the power of such an approach for 2D images. They have shown how image composition can be used with sophisticated, spatially-varying light mattes to create compelling technical illustrations or composite photographs from a set of photographs with different, locally inconsistent, lighting.

In this dissertation we explore how the use of inconsistent lighting in 3D visualization may allow us to convey a better perception of geometry than consistent lighting. We also explore how we could automatically determine lighting parameters

to effectively convey a large number of data features such as local surface orientation, curvature, silhouettes, and fine details.

1.3.2 Our Approach

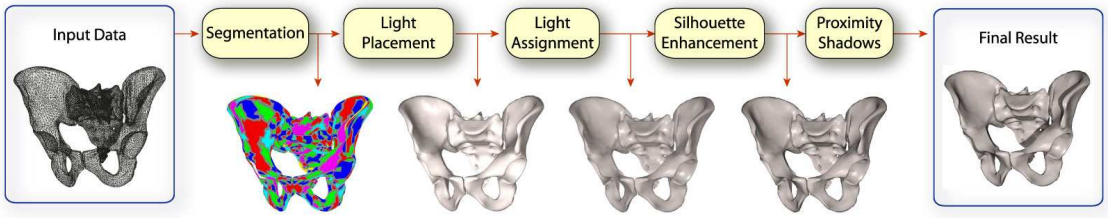


Figure 1.1: *Overview of our lighting design pipeline: The input model is segmented using a curvature-based-watershed method into a set of patches. The light placement function models the appropriateness of light directions for illuminating the model. This is done by using the curvature-based segmentation as well as the diffuse and specular illumination at every vertex. Lights are placed and assigned to patches based on the light placement function. Silhouette lighting and proximity shadows are added for feature enhancement.*

We introduce the framework of *geometry-dependent lighting* [56] that allows lighting parameters to be defined independently and possibly inconsistently over an object or scene based on the local geometry. We present and discuss *Light Collages* [55, 56], a lighting design system with geometry-dependent lights for effective feature-enhanced visualization. Our algorithm segments the objects into local surface patches and places lights that are locally consistent but globally discrepant to enhance the perception of shape. Our Light Collages framework designs the lighting so that the diffuse illumination is proportional to the local curvature and specular highlights are only on highly curved regions. This helps elucidate geometry details. Our system casts proximity shadows to illustrate the depth relationship between two adjacent regions in the rendered image. To make the objects stand out from their

background, the system enhances silhouettes by producing a dark silhouette for a bright background and a bright silhouette for a dark background. Figure 1.1 shows the overview of our approach. We use spherical harmonics for efficiently storing and computing light placement and assignment. We also outline a method to find the minimal number of light sources sufficient to illuminate an object well with our globally discrepant lighting approach.

The main contributions of this dissertation on lighting design are:

- *Globally Inconsistent Lighting:* We present a framework of local illumination that relaxes the constraint of globally consistent lighting and show how it can generate comprehensible renderings.

- *Lighting Design:* We discuss automatic placement of multiple light sources to enhance view-dependent visualization.

- *Feature Enhancement:* Silhouettes and shadows add important details in technical illustrations. Here we show how silhouette and shadow lighting can be integrated into a local illumination framework for enhancing features in scientific datasets.

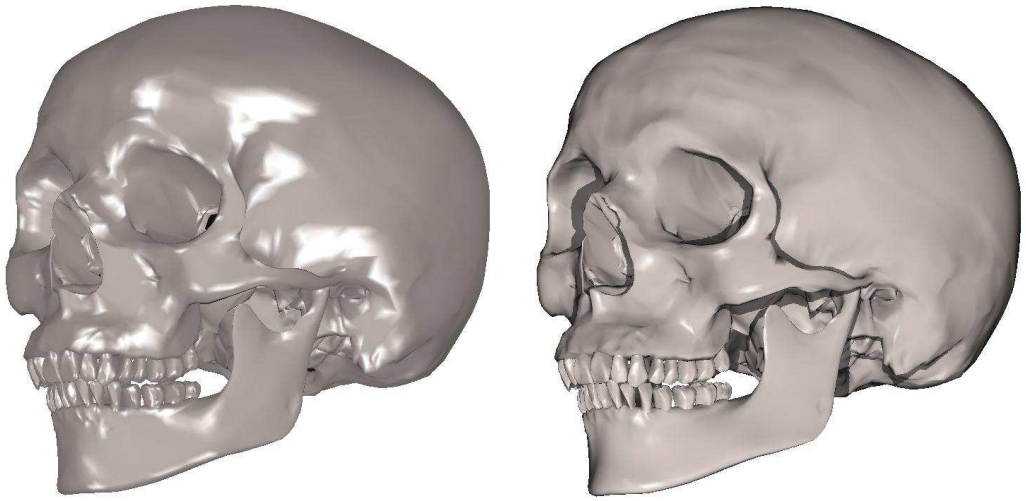
- *Efficient Computation:* We have improved the run-time efficiency of our system by a factor of 20 and reducing the memory footprint by over two orders of magnitude. We discuss how one can achieve this by using a spherical-harmonic-basis representation for light placement and assignment.

- *Minimality of the Light Sources:* The benefits of adding more discrepant lights

diminish with the total number of lights in the system. Another novel contribution of our work is the notion of minimality of lights for a given view and geometry and showing how this changes with simplifications of the geometry.

We explore further detail of how the geometry-dependent lighting framework and its implementation determine lighting parameters for enhancing visualization in Sections 2.2–2.4, and how the techniques for feature enhancement and efficient computation help to elucidate local geometric details effectively and efficiently in Section 2.5, and Sections 2.6 and 2.7.

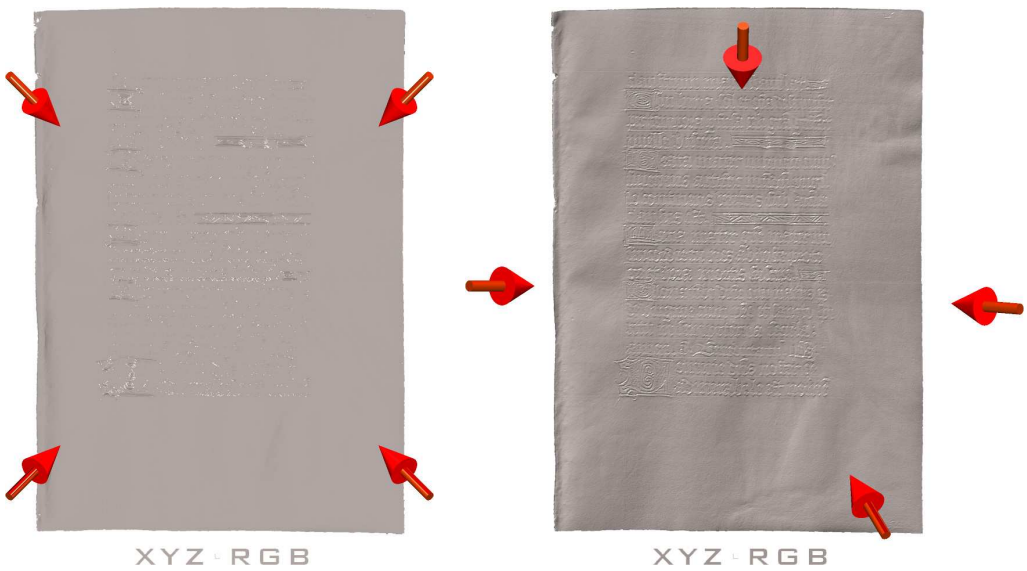
1.3.3 Summary of Lighting Design Results



(a) (b)

Figure 1.2: *(a) Consistent lighting with four lights at the front four vertices of a cube, and (b) a Light Collage rendering with 4 lights. Material properties are the same for both renderings.*

Figures 1.2 and 1.3 show the rendering results using our Light Collages system. In Figure 1.2 (a), we show the result of traditional lighting on the skull model with



(a) (b)

Figure 1.3: *Light Collages for the Rouen Manuscript: (a) Rendered by four consistent lights arranged at the front four corners of a cube, and (b) Rendered by Light Collages with four lights using* 25 *SH coefficients.*

four light sources placed at the front four vertices of a cube. Figure 1.2 (b) shows the same skull model rendered by Light Collages with four automatically-placed lights. We have used the same lighting and material properties for generating the two images. As you can see in (a), the specular highlight from consistent lighting will sometimes cause large bright areas on flat regions, while highlights from our methods (b) are only on highly curved regions. This helps elucidate geometry details. The proximity shadow cast by the upper cheek bone in Figures 1.2 (b) nicely illustrates the depth relationship between these two regions of the skull. Naive consistent lighting as shown in Figures 1.3 (a) fails to capture the fine details of the characters and the subtle variations and wrinkles in the manuscript. Figure 1.3(b) nicely shows these subtle variations in the geometry with our geometry-dependent discrepant lighting.

We use spherical harmonics for efficiently storing and computing light placement and assignment. Table 1.1 shows how much we could improve the computation-time efficiency through the spherical-harmonic representation compared with a direct computation at $12K$ uniformly-distributed directions. As one can see, the spherical-harmonic method with 5 bands is almost 20 times faster than the direct computation. Further, since we only need to store 25 spherical-harmonic coefficients per vertex instead of over $12K$ directional samples, our spherical-harmonic-based lighting design approach reduces the required memory by a factor of over 500.

Table 1.1: Run Times for Light Collages

Model	**Skull (33K verts)**	**Pelvis (17K verts)**
SH Bands	**Time (sec)**	**Time (sec)**
2	4.04	2.54
3	5.30	3.41
4	8.05	5.19
5	13.42	7.11
6	19.85	12.05
7	29.58	16.62
8	42.81	23.63
9	60.36	35.51
10	81.30	43.72
Direct Computation	234.17	138.82

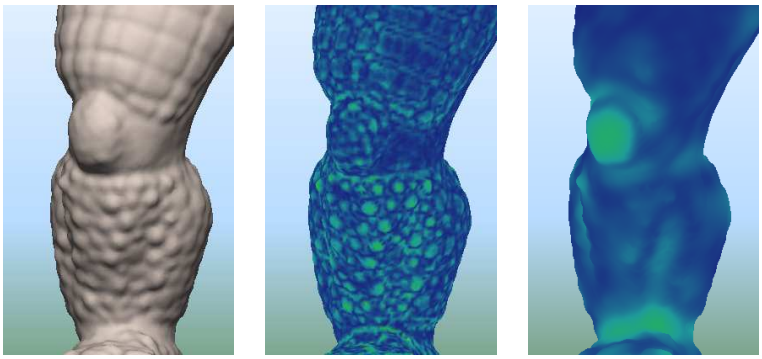
1.4 Mesh Saliency

1.4.1 Problem Definition

We have witnessed significant advances in the theory and practice of 3D graphics meshes over the last decade. These advances include efficient and progressive representation [38, 47], analysis [53, 64, 93], transmission [4], and rendering [59] of

very large meshes. Much of this work has focussed on using mathematical measures of shape, such as curvature. The rapid growth in the number and quality of graphics meshes and their ubiquitous use in a large number of human-centered visual computing applications, suggest the need for incorporating insights from human perception into mesh processing. Although excellent work has been done in incorporating principles of perception in managing level of detail for rendering meshes [58, 78, 99], there has been less attention paid to the use of perception-inspired metrics for processing of meshes. Our goal in this dissertation is to bring perception-based metrics to bear on the problem of processing and viewing 3D meshes.

1.4.2 Our Approach



(a) (b) (c)

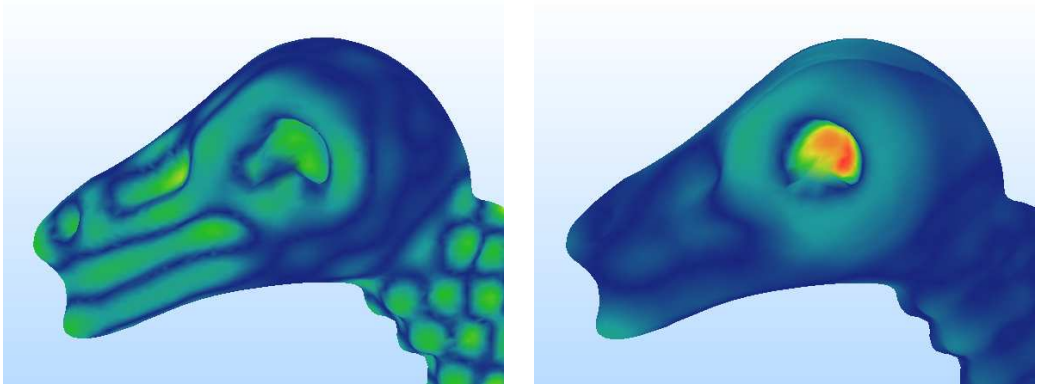
Figure 1.4: *Curvature alone is inadequate for assessing saliency since it does not adequately consider the local context. Image (a) shows a part of the right leg of the Stanford Armadillo model. Image (b) visualizes the magnitude of mean curvatures and (c) shows our saliency values. While (b) captures repeated textures and fails to capture the knee, (c) successfully highlights the knee.*

Purely geometric measures of shape such as curvature have a rich history of use in the mesh processing literature. For instance, Heckbert and Garland [36] show that their quadric error metric is directly related to the surface curvature.

Mesh simplifications resulting from minimizing the quadric errors result in provably optimum aspect ratio of triangles in the L_2 norm, as the triangle areas approach zero. However, a purely curvature-based metric may not necessarily be a good metric of perceptual importance. For example, a high-curvature spike in the middle of a largely flat region will be likely perceived to be important. However, it is also likely that a flat region in the middle of densely repeated high-curvature bumps will be perceived to be important as well. Repeated patterns, even if high in curvature, are visually monotonous. It is the unusual or unexpected that delights and interests. As an example, the textured region with repeated bumps in the leg of the Armadillo shown in Figure 1.4(a) is arguably visually less interesting than an isolated but smooth feature such as its knee (Figure 1.4(c)).

In this dissertation, we introduce the concept of *mesh saliency* [57], a measure of regional importance, for 3D meshes, and present a method to compute it. Our method to compute mesh saliency uses a center-surround mechanism. We use the center-surround mechanism because it has the intuitive appeal of being able to identify regions that are different from their surrounding context. We are also encouraged by the success of these mechanisms on 2D problems.

We expect a good model of saliency to operate at multiple scales, since what is interesting at one scale need not remain so at a different scale. A good saliency map should capture the interesting features at all perceptually meaningful scales. Figure 1.5(a) shows a saliency map at a fine scale where small features such as the nose and mouth have high saliency, while a saliency map at a larger scale (Figure 1.5(b)) shows the eye to have a higher saliency. We use these observations to



(a) (b)

Figure 1.5: *Saliency is relative to the scale. Image (a) shows the saliency map of the Cyberware Dinosaur head at a small scale, and image (b) shows the map of its saliency at a larger scale. In image (a), the small-scale saliency highlights the small features such as nose and mouth and in image (b), the large-scale saliency identifies a larger feature such as the eye.*

define mesh saliency in a scale-dependent manner using a center-surround operator on Gaussian-weighted mean curvatures. We observe that such a definition of mesh saliency is able to capture what most would classify as visually interesting regions on a mesh. A number of tasks in graphics can benefit from a computational model of mesh saliency. In this dissertation we explore the application of mesh saliency to mesh simplification, view selection, and lighting in Sections 3.3 and 3.4, and Chapter 4.

The main contributions of this dissertation on mesh saliency are:

1. **Saliency Computation:** There can be a number of definitions of saliency for meshes. We outline one such method for graphics meshes based on the Gaussian-weighted center-surround evaluation of surface curvatures. Our method has given us very promising results on several 3D meshes.

2. **Salient Simplification:** We discuss how traditional mesh simplification meth-

(a) (b)

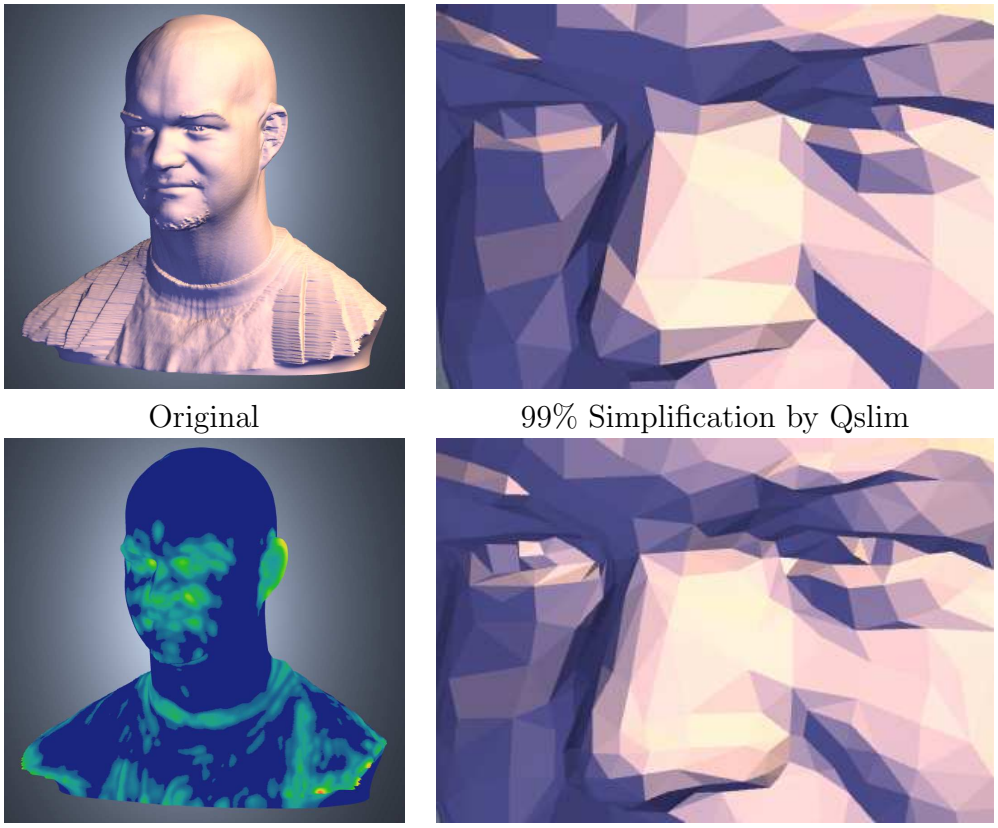
Figure 1.6: *Mesh Saliency: Image (a) shows the Stanford Armadillo model, and image (b) shows its mesh saliency.*

ods can be modified to accommodate saliency in the simplification process. Our results show that saliency-guided simplification can easily preserve visually salient regions in meshes that conventional simplification methods typically do not.

3. **Salient Viewpoint Selection:** As databases of 3D models evolve to very large collections, it becomes important to automatically select viewpoints that capture the most salient attributes of objects. We present a saliency-guided method for viewpoint selection that maximizes visible saliency.

4. **Salient Lighting:** Lighting has long been used in art and scientific illustrations to enhance our perception of shape as well as important features. We discuss a saliency-guided modification of the lighting pipeline to emphasize salient regions.

We foresee the computation and use of mesh saliency as an increasingly im-



Original | 99% Simplification by Qslim

Saliency | 99% Simplification guided by saliency

Figure 1.7: *99% Simplification of the human face*

portant area in 3D graphics. As we engage in image synthesis and analysis for ever larger graphics datasets and as the gap between processing capabilities and memory-access times grows ever wider, the need for prioritizing and selectively processing graphics datasets will increase. Saliency can provide an effective tool to help achieve this.

1.4.3 Summary of Mesh Saliency Results

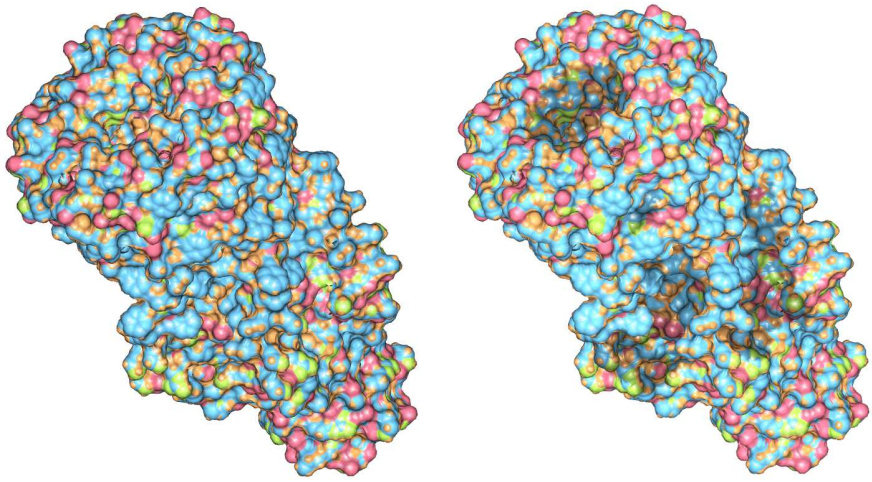
Figure 1.6 shows the mesh saliency for the Stanford Armadillo model. Our approach for computing saliency classifies repeating patterns as non-salient. Therefore, high-curvature regions such as the repeated bumps on the legs of the Armadillo

(a) (b)

Figure 1.8: *Image (a) shows a viewpoint selected by maximizing visible saliency, and image (b) shows a viewpoint selected by maximizing visible mean curvature. Since saliency negates the repeated hair texture in image (a), the method based on saliency selects the more interesting region of face instead of the top of the head.*

model in Figure 1.6 have low saliency. David's hair in Figure 1.8 also shows an example of repeating patterns with a high curvature and a low saliency.

We have shown the applicability of mesh saliency to several tasks in graphics including mesh simplifications, viewpoint selection, and lighting design. Figure 1.7 shows the result of our saliency-guided simplification method. We have modified Qslim [25] by multiplying mesh saliency to the quadric errors for rescheduling the edge contractions. Figure 1.7 shows that our method preserves more triangles on the interesting regions such as eyes, nose, mouse, and ears of the human face model. Figure 1.8 shows that our saliency-based method selects a more pleasing view for the David's head model than a curvature-based method. We have also used saliency for emphasizing salient regions by using varying illumination parameters based-on



(a) (b)

Figure 1.9: *Rendering of Ecoli membrane channel: (a) shows a rendering with conventional lighting, and (b) shows a saliency-guided rendering. Our saliency-guided rendering clearly shows the central channel as well as the oblique clefts on the side which are not shown with traditional local illumination. Data for this channel is courtesy S. Sukharev [92].*

the saliency. In Figure 1.9, our salient lighting shows the central channel as well as the oblique clefts on the side of the Ecoli membrane channel.

1.5 Outline of the Dissertation

The remainder of this dissertation is organized as follows. Chapter 2 describes how we use the framework of geometry-dependent lighting for effectively visualizing the shape of 3D objects. We describe detailed algorithms of *Light Collages* system as well as techniques for efficient computation. In Chapter 3, we introduce our perception-inspired metric, *mesh saliency*, as a measure of regional importance for graphics meshes. We discuss how mesh saliency can be incorporated into graphics applications such as mesh simplification and viewpoint selection and present examples that show visually appealing results from using mesh saliency. Chapter 4

describes our approach to change the lighting guided by saliency for focusing the user's attention on salient regions by illumination-based emphasis. Finally, Chapter 5 suggests several ideas for future directions.

Chapter 2

Geometry-dependent Lighting

Artists and illustrators enhance perception of features with lighting that is locally consistent and globally inconsistent. Recent research by Ostrovsky *et al.* [67] suggests that our brain perceives the shape-from-shading cues locally and does not use large regions of the visual field for shape-from-shading analysis. Inspired by these techniques, we introduce *geometry-dependent lighting* that allows lighting parameters to be defined independently and possibly inconsistently over an object or scene based on the local geometry. We present and discuss *Light Collages*, a lighting design system with geometry-dependent lights for effective feature-enhanced visualization. Our algorithm segments the objects into local surface patches and places lights that are locally consistent but globally discrepant to enhance the perception of shape. We further enhance the perception of features using silhouette enhancement and proximal depth shadows. We use spherical harmonics for efficiently storing and computing light placement and assignment. We also outline a method to find the minimal number of light sources sufficient to illuminate an object well with our globally discrepant lighting approach.

2.1 Previous and Related Work

In photography, cinematography, and stage lighting, the specification of light position, direction, color, intensity, and type determines the appearance of the resulting scene. Kahrs *et al.* [44] have summarized the lighting design approaches for computer animation. They distinguish between logical and pictorial lights. *Logical* lights are motivated by actual sources of light in a scene that the viewer can see or imply. For example, the *key* light is used in a scene as the primary source of illumination. In addition to logical lighting cinematographers use *pictorial* lighting for enhancing the artistic aesthetics of the scene. For example, *back or rim* lights are used to separate the object from the background, and *fill* lights are used to soften and fill the shadows.

Much of the current work on lighting design in 3D graphics and visualization has focused on determining the parameters for logical lights and has generally overlooked pictorial lighting. We classify the lighting design methods for graphics as either *direct* or *indirect*. Conventional lighting design methods are direct – they require a user to directly specify the lighting parameters. The user starts out by specifying an initial set of lighting parameters and then visually evaluates the results. The lighting parameters are then changed iteratively till the graphics rendering converges to a desired output. Although the visual results from using a direct light specification may be satisfactory, the process itself leaves much to be desired. First, direct lighting design is often iterative and time consuming. Second, it requires a significant expertise on the part of the user to achieve desired visual effects

from light placement, such as locations of highlights and shadows. The approach of Design Galleries [62] addresses these shortcomings by using several user-specified lighting parameters (excluding light placement), generating a set of renderings with randomly placed lights, and having a user browse and hierarchically select the renderings that are desirable. The LightKit system [33] allows a user to interactively adjust lighting to enhance visualization. This system allows camera-relative lights that include a dominant light, headlights, and backlights. The system also allows the user to adjust the light color and warmth of lighting.

Indirect lighting design methods use scene properties that are either specified by a user or procedurally estimated. In *user-specified indirect lighting design*, the user specifies the desired highlights or shadows and the system then infers the light placement to achieve them [15, 49, 70, 71, 82]. In *procedural indirect lighting design*, the system automatically infers light placement and parameters by optimizing a set of perceptual criteria for a given view. Shacked and Lischinski [83] derive light placement for up to two light sources by optimizing a perception-based image quality objective function. Their objective function includes six terms that are based on shading gradients, pixel luminance statistics, and illumination direction. Gumhold [31] has developed a light-placement strategy by maximizing a perceptual entropy objective function as measured from a rendered image.

Although we have not come across prior work on physically-inconsistent lighting design for 3D graphics and visualization, there is a sizable literature on physically-implausible lighting models collectively referred as *non-photorealistic lighting*. Gooch *et al.* [28] have developed a lighting model that uses luminance and changes in hue to

convey surface orientation, edges, and highlights. Sousa *et al.* [87] have incorporated lighting into adaptive pen-and-ink stroke lengths to convey shape. Hamel [34] has developed a lighting model that incorporates five components – standard lighting with shadows, rim shadow lighting, curvature shading, transparency, and volume illumination. Sloan *et al.* [85] have developed an effective method to transfer the shading from one object to another using a sphere (environment map) as an intermediary. Anderson and Levoy [5] have used curvature- and accessibility-based shading [66] to enhance the visualization of cuneiform tablets. Vicinity lighting [88] improves upon the idea of accessibility shading by using uniform diffuse lighting and occlusion by local occluders. Akers *et al.* [3] have used image composition with sophisticated, spatially-varying light mattes to create compelling technical illustrations from a set of photographs taken under different lighting conditions.

To the best of our knowledge, none of the previous work has tried to render the same object with multiple light sources with each light source lighting a different region of the object. In fact, the general advice seems to have been to illuminate objects with a single light placed above and to the left of the object [90]. In this chapter we discuss the idea of geometry-dependent lighting that involves lighting different regions of an object with multiple light sources to render it in a more visually comprehensible manner, while retaining its traditional 3D-graphics-rendered look and feel. Our goal is to provide effective visualization conveying a large number of features such as local surface orientation, curvature, silhouettes, and fine texture.

2.2 Light Collages Overview

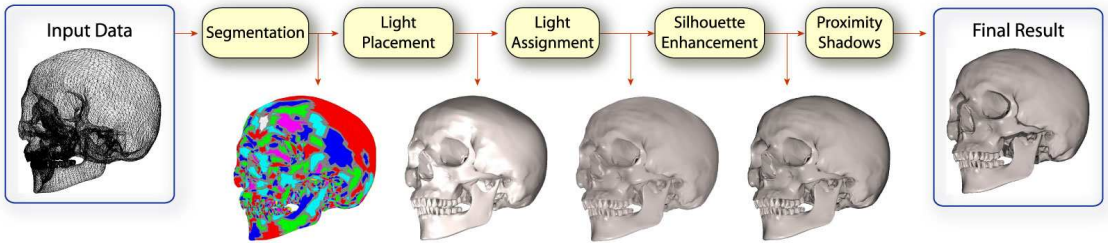


Figure 2.1: *Overview of our lighting design pipeline: The input model is segmented using a curvature-based-watershed method into a set of patches. The light placement function models the appropriateness of light directions for illuminating the model. This is done by using the curvature-based segmentation as well as the diffuse and specular illumination at every vertex. Lights are placed and assigned to patches based on the light placement function. Silhouette lighting and proximity shadows are added for feature enhancement.*

The geometry-dependent lighting framework allows local regions to be illuminated by discrepant lights based on their local geometry. Our Light Collages system automatically designs geometry-dependent lighting for a given view by placing directional light sources and assigning them to different regions of an object. Let us define the problem more formally. Consider an object composed of n surface patches $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$. Let there be a set of m unknown light sources $\mathcal{L} = \{l_1, l_2, \ldots, l_m\}$. The problem we solve here is: *Given $\mathcal{P}$, m, and a viewer position, generate $\mathcal{L}$ and a mapping $\mathcal{M}$ that pairs each light $l_i \in \mathcal{L}, 1 \leq i \leq m$ to a subset of patches $\mathcal{P}_i \subset \mathcal{P}$ that it lights, to best elucidate the local structure of the object.* Here, the subsets $\mathcal{P}_i$ are mutually exclusive and collectively exhaustive: $\mathcal{P}_i \cap \mathcal{P}_j = \emptyset, 1 \leq i, j \leq m$, $\bigcup_{i=1}^{m} \mathcal{P}_i = \mathcal{P}$. Then each patch $p_j \in \mathcal{P}_i$ is assigned a primary light source $l_i = \mathcal{M}(p_j)$. We believe that the idea of *best elucidation* is open to interpretation. There is strong evidence that conveying the local curvature

information is important in shape perception. Girshick *et al.* [26] present several compelling visual examples that show that placing line strokes along principal directions of curvature are more effective than other directions. Additionally, user studies on light source placement by Gumhold [31] have indicated that observers tend to select light source directions that favor surface curvature elucidation.

The Light Collages system first segments the input model into a set of patches, then places lights and assigns them to patches, and finally adds silhouette lighting and proximity shadows for feature enhancement as illustrated in Figure 2.1. In the sections 2.3–2.5, we discuss each stage of the Light Collages pipeline in detail.

2.3 Surface Segmentation

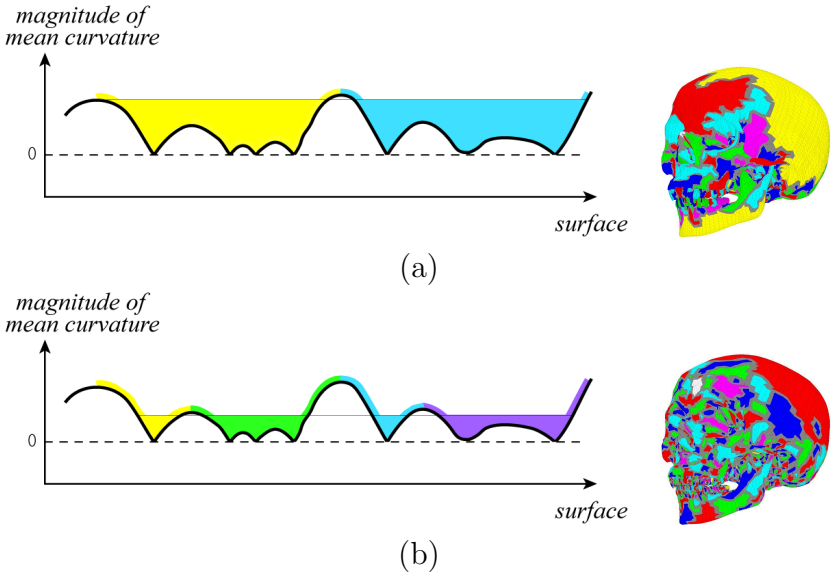
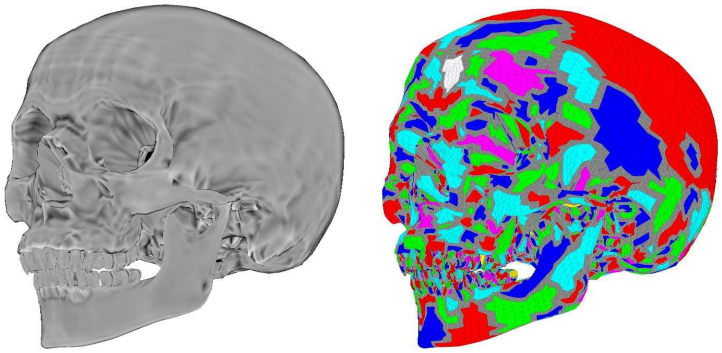


Figure 2.2: *The watershed algorithm: First, we assign unique labels (patch IDs) to local minimums. Next, imagine that we place a drop of water at a vertex that will flow to the local curvature minimum. We assign the label of the local minimum to the vertex where the water drop was placed. Figure shows (a) coarse and (b) fine segmentation. For all models used in this dissertation, we use 7.5% of the range of curvature difference as a threshold.*

We segment the input model into a set of patches to define the local regions which will be lighted discrepantly. The segmentation of an object is a classical area of research in computer vision and image processing. Any of the vast number of segmentation algorithms can be used for object segmentation at this stage depending on what the goals of the segmentation-based lighting design are. In this dissertation, we segment the object into patches based on local curvature. The goal is to make each patch be a collection of triangles with similar curvature values.

We first compute the mean curvature at each vertex of the input mesh as the average of its two principal curvatures, which are computed using Taubin's method [93]. Then we segment using a simple watershed algorithm based on Mangan and Whitaker's method [60]. First, their method finds vertices with local curvature minima and uses each of them as a seed for growing a new patch. The method then iteratively assigns vertices to these patches. A path of steepest descent is computed from each unassigned vertex till it reaches a seed vertex with a local curvature minimum. The vertex is assigned to the patch corresponding to this seed vertex. A *watershed depth* is computed for each patch based on the minimum difference in curvature values between a boundary vertex and the seed vertex for that patch. Patches whose watershed depth is below a threshold depth are merged. Figure 2.2 illustrates how the segmentation can be decreased or increased by respectively raising or lowering the threshold depth as shown in Figures 2.2 (a) and (b). Figure 2.3(a) shows the distribution of the curvature over the Skull model and Figure 2.3(b) shows the results of our segmentation of the object into multiple surface patches: $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$.



(a) (b)

Figure 2.3: *In (a) we show curvature distribution over the Skull model. Convex regions are shown brighter and concave regions are darker. Figure (b) shows the results of our curvature-based segmentation.*

2.4 Procedural Lighting Design

We assume that all the lights are white and directional. Our lighting design algorithm proceeds in two interleaved phases. In one phase we identify the placement of a light and in the other phase we assign the light to appropriate patches.

2.4.1 Light Placement Function

Curvature influences the illumination gradient across a surface and is an important visual cue to shape. We use a combination of local lighting models to enhance the appearance of high-curvature areas of an object from a given viewpoint. A specular highlight on a shiny surface can easily vanish with even small perturbations of the viewing direction, surface normal, or light direction. For a low-curvature area, the specular highlight hides the subtle geometric changes because of over exposure. However, for a region with high curvature, the specular highlight is useful as it can result in a sharp curvature-based highlight, and thus help illustrate object detail.

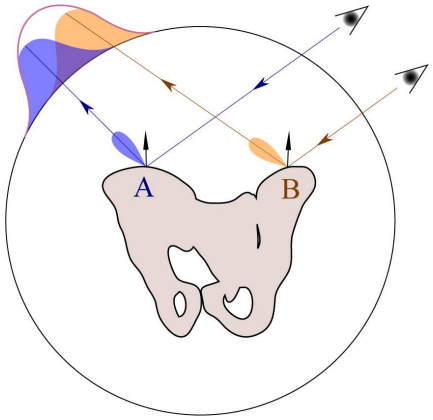


Figure 2.4: *A view-dependent weight function for each surface point is added to the light placement function defined in the directional space (shown here by the large circle). The light placement function models the appropriateness of placing a light along a direction.*

As an example, let us consider two points A and B on which we would like to place specular highlights (Figure 2.4). If we have the freedom to place a directional light source along any direction, we would like to place it in a direction that maximizes the possibility of having highlights on points A and B. We can infer the light directions that will cause specular highlights to appear on points A and B by using the view direction, the shape, and the material properties of an object. Using the reciprocity principal, this is equivalent to shooting a ray of light from the viewpoint to the points A and B, and having that light specularly reflect out to the environment. The specularly reflected rays will result in a distribution around the direction of mirror reflection as shown in Figure 2.4. The blue and orange blobs on the upper left region of the circle represent the probability density function (PDF) of the reflected ray along those directions. The total probability of a specular highlight can be computed by the sum of the individual PDFs, as shown by the purple curve. Thus, following the reciprocity principal, if we were to place a light source

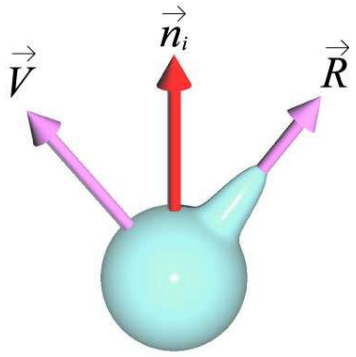


Figure 2.5: *Specular weight function $S(i, \vec{l})$ is defined as the fall-off function around the reflection vector $\vec{R}$ weighted by curvature.*

in the direction where the purple curve has the largest value, we would get the best highlights at both the points A and B for the given view position.

We extend the above ideas to define a light placement function $P(\overrightarrow{l})$ that models the appropriateness of placing a light in the direction $\overrightarrow{l}$. Such a light placement function should include contributions from both specular as well as diffuse illumination. Let $\mathcal{P}$ be the set of surface patches for an object. Let $\overrightarrow{v}$ be the view vector, $\overrightarrow{l}$ be the light direction, and $\overrightarrow{h}$ be the halfway unit vector along the direction $\overrightarrow{l} + \overrightarrow{v}$. Further, let κ_i be the mean curvature, $\overrightarrow{n_i}$ be the normal vector, and $\vec{R}$ be the reflection of viewing direction $\vec{v}$ about the normal $\overrightarrow{n_i}$ at a vertex i on the surface. We define the specular weight function S for the vertex i with a shininess s as: $S(i, \overrightarrow{l}) =\mid \kappa_i \mid (\overrightarrow{n_i} \cdot \overrightarrow{h})^s$. Figure 2.5 shows the specular weight function.

Given a view direction, we compute $S(i, \overrightarrow{l})$ for each vertex i and for a set of uniformly-distributed light directions $\overrightarrow{l}$. In our implementation we use $12K$ uniformly-distributed directions $\overrightarrow{l}$. However, the use of specular highlights alone is not desirable, as shown by Gumhold [31]. In addition to specular lighting, we would also like to consider diffuse lighting. We have designed the diffuse lighting

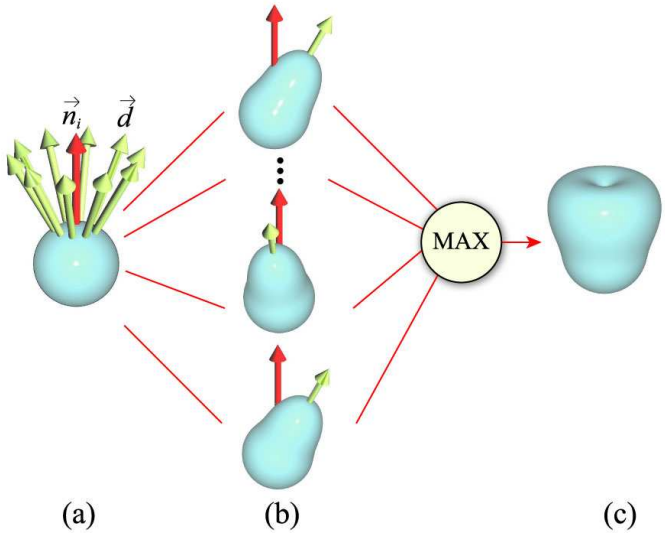
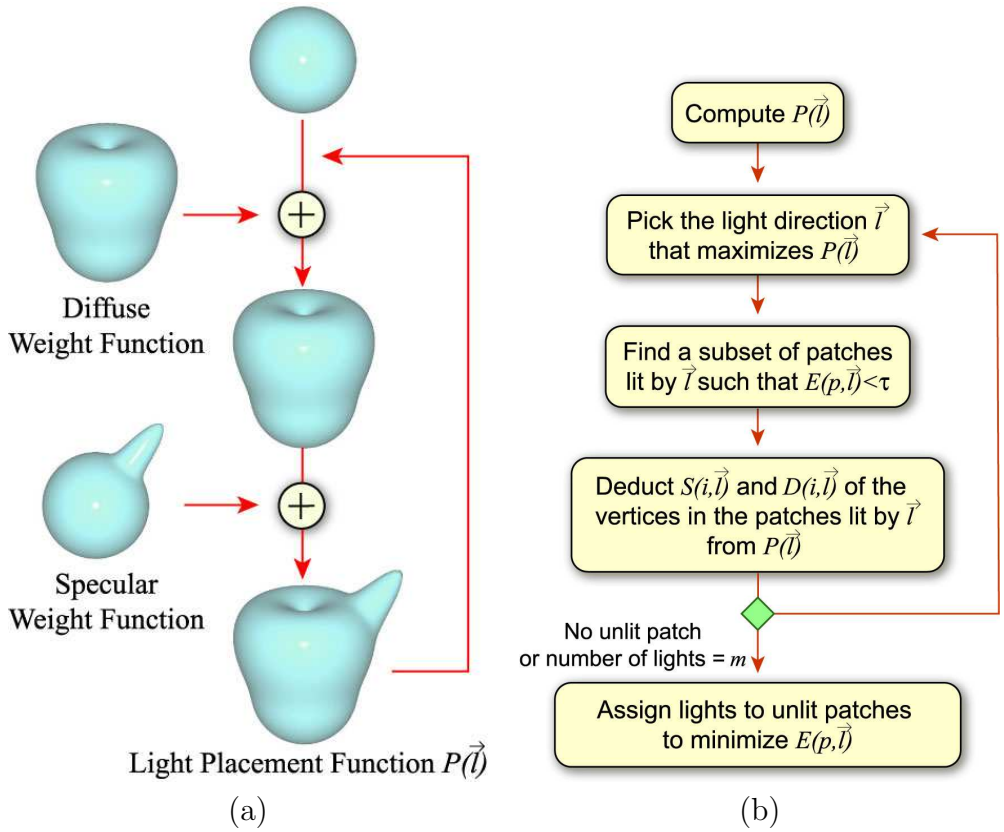


Figure 2.6: *Computation of diffuse weight function for a vertex with normal $\vec{n}_i$ and curvature intensity c_i: First, (a) we define the set of light directions $\vec{d} \in \mathcal{D}$ for which $\vec{n}_i \cdot \vec{d} = c_i$. These directions $\vec{d}$ are shown by green arrows. Figure (b) shows the cosine fall-off for each $\vec{d} \in \mathcal{D}$. (c) The diffuse weight function $D(i, \vec{l})$ is the upper envelope (maximum) of the functions shown in Figure (b).*

component of the light placement function to adapt to the local curvature on a patch-by-patch basis. Figure 2.3(a) shows curvature distribution over the Skull model. We define the *curvature intensity* c_i at a vertex i to be its normalized mean curvature, i.e. $c_i = (\kappa_i - \kappa_{min})/(\kappa_{max} - \kappa_{min})$, where κ_i is the mean curvature at vertex i, and κ_{max} and κ_{min} are the maximum and minimum values of the mean curvature among all the vertices of the input mesh, respectively. For a vertex i with normal vector $\overrightarrow{n_i}$, let $\mathcal{D}$ be the set of light directions whose diffuse color is same as the curvature intensity c_i: $\mathcal{D} = \{ \overrightarrow{d} \,|\, \overrightarrow{d} \cdot \overrightarrow{n_i} = c_i \}$.

We define the diffuse weight function $D(i, \overrightarrow{l})$ for vertex i in the direction of $\overrightarrow{l}$ such that the diffuse illumination at vertex i is similar to the curvature intensity c_i. We compute it as the upper envelope (maximum) of the dot product between $\overrightarrow{l}$ and all $\overrightarrow{d} \in \mathcal{D}$ as: $D(i, \overrightarrow{l}) = \underset{\overrightarrow{d} \in \mathcal{D}}{Max} \overrightarrow{l} \cdot \overrightarrow{d}$. This is shown in Figure 2.6.



(a) (b)

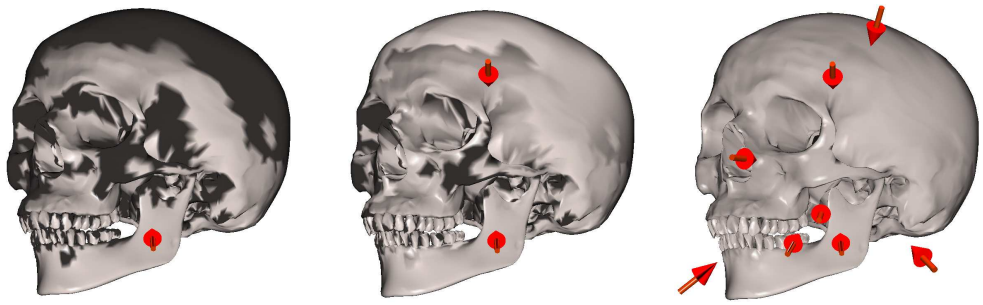
Figure 2.7: *The light placement function $P(\vec{l})$ is computed in Figure (a) by adding diffuse and specular weight functions. Figure (b) shows the flowchart of the process for light placement and assignment.*

The light placement function can be computed as the sum of specular and diffuse weight functions over all surface points. For any light direction $\overrightarrow{l}$ the value of the light placement function $P(\overrightarrow{l})$ along that direction is given by:

$$P(\overrightarrow{l}) = \sum_i (S(i, \overrightarrow{l}) + D(i, \overrightarrow{l}))$$

2.4.2 Light Placement and Assignment

We select the best m lights $\mathcal{L} = \{l_1, l_2, ..., l_m\}$ by using the light placement function $P(\overrightarrow{l})$, as follows. We identify the light direction $\overrightarrow{l}$ that maximizes $P(\overrightarrow{l})$.



(a) (b) (c)

Figure 2.8: *Partial surface lighting with (a) first light, and (b) first two lights, and (c) eight lights. The red arrows show the light directions. The dark regions in (a) and (b) are the patches not lighted by the current partial lights. No blending is used here.*

We select this to be the direction of the first light l_1. We then identify the patches which will be lighted by the light l_1. For any light $l_k \in \mathcal{L}$ and patch $p \in \mathcal{P}$, let $\mathcal{S}_p$ be the set of points that are on p and let $I_i(l_k)$ be the illuminated intensity at vertex $i \in \mathcal{S}_p$ due to light l_k. We define a function $E(p, l_k)$ that measures the similarity of the illuminated intensity $I_i(l_k)$ for vertices i in the patch p to its curvature intensity as:

$$E(p, l_k) = \sum_{i \in \mathcal{S}_p} (I_i(l_k) - c_i)^2$$

For the first light l_1, we assign l_1 to a patch $p \in \mathcal{P}$ whenever $E(p, l_1)$ is less than a threshold τ (currently we use $\tau = 0.15$), i.e. $\mathcal{M}(p) = l_1$. We deduct the contributions of the vertices in the patches lighted by this light l_1 from the light placement function. We repeat this process until m lights are selected. For each unlit patch, the light l_k which minimizes $E(p, l_k)$ is assigned to p : $\mathcal{M}(p) = \arg\min_{l_k \in \mathcal{L}} E(p, l_k)$. Figures 2.8 (a)–(c) show the lighting with one, two, and eight lights. Patches that are not lighted are shown dark without any blending with the neighboring patches.

2.4.3 Illumination Blending

Our Light Collages framework allows patches to be assigned different lights even though the patches are adjacent. A straightforward implementation of this idea might result in sharp visual discontinuities across patch boundaries that are lighted differently. Such shading discontinuities are disconcerting especially when they occur in absence of shape discontinuities. To alleviate such visual artifacts we blend illumination from neighboring patches. As mentioned earlier, every vertex i in a patch p_j is illuminated by light $\mathcal{M}(p_j)$. The blended illumination at a vertex i is a weighted sum of illuminations from the primary lights for all the patches $\mathcal{N}_j$ that are next to p_j: $\mathcal{N}_j = \{p_k \mid \partial p_j \cap \partial p_k \neq \emptyset\}$, where ∂p_j denotes the boundary of patch p_j. Let the primary light for patch $p_k \in N_j$ be given by $l_k = \mathcal{M}(p_k)$. Let the weight of vertex i with respect to the primary light of patch p_k be based on the distance function $d()$ of vertex i from the boundary ∂p_k and be given by:

$$w_{ik} \propto \frac{1}{1 + d(i, \partial p_k)}$$

We define the distance $d(i, \partial p_j)$ to be zero for a vertex inside or on the boundary of the patch p_j. Therefore, the weight of a vertex i inside patch p_j is $w_{ij} = 1$. The distribution of the blending weights at vertices around a patch is shown in Figure 2.9.

A simple weighted sum of illuminations may increase the overall brightness which tends to result in diminishing the visual discriminability amongst object features. To balance the rendering brightness we normalize the illumination with the

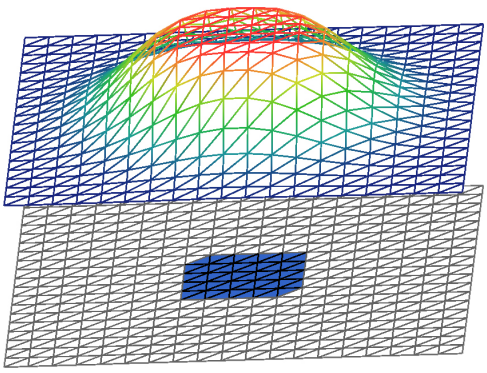


Figure 2.9: *Figure shows the weights for blending illumination. The lower mesh shows the patch (in blue) and its neighborhood. For each vertex of the lower mesh, the vertex of the upper mesh vertically above it represents the blending weight. Note that the weight stays constant over the patch and then gradually falls off.*

blending weights for a given vertex. Let the illumination at vertex i due to light l_k be given by $I_i(l_k)$ as defined in Section 2.4.2. Then, the final illumination formula for a vertex i in patch p_j with neighbors $\mathcal{N}_j$ is given by:

$$\overline{I_i} = \frac{\sum_k w_{ik} I_i(l_k)}{\sum_k w_{ik}}, p_k \in \mathcal{N}_j, l_k = \mathcal{M}(p_k)$$

2.5 Feature Enhancement

2.5.1 Silhouette Enhancement

Usually silhouettes characterize large depth discontinuities. Therefore, a well-defined silhouette makes an object easier to comprehend by making it more easily distinguishable from its surroundings. Cinematographers use backlights for separating the foreground from the background. They traditionally place backlights behind an object to generate a thin rim of light around its silhouette. Backlights are also called rim, hair, or separation lights. In particular, the lights at the three-quarters-

(a) (b) (c)

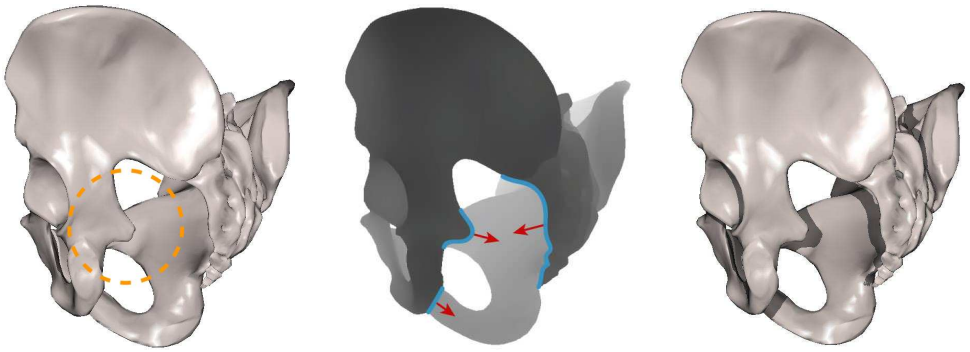
Figure 2.10: *Rendering (a) without, and (b), (c) with silhouette lighting.* $(1 - \vec{n}_i \cdot \vec{v})^u$ *is used as the silhouette light's weight factor. (b) and (c) show silhouette enhancement with* $u = 4$ *and* $u = 2$ *respectively.*

back position are called as kicker lights [44].

To distinguish an object from its background, we produce a dark silhouette for a bright background and a bright silhouette for a dark background. We use a simple fall-off formula weighted by $\omega_s = (1 - \overrightarrow{n_i} \cdot \overrightarrow{v})^u$, for adding an additional silhouette light at vertex i with normal $\overrightarrow{n_i}$ and view direction $\overrightarrow{v}$. The results of incorporating black silhouette lighting appear in Figure 2.10. We compute the silhouette-enhanced illumination as the linear blend of the silhouette lighting H_i weighted by ω_s and the existing illumination: $(1 - \omega_s)I_i + \omega_s H_i$.

2.5.2 Proximity Shadows

Perception of depth through carefully placed shadows is an important visual cue for comprehending the spatial relationships between objects. As an example, it may be difficult to distinguish two surface patches if they have similar illumination but different distances from the viewer and partially overlap in space as seen by the viewer. However, if the front patch casts a visible shadow on the other patch, their



(a) (b) (c)

Figure 2.11: *Figure (a) shows a pelvis model rendered without proximity shadows. The illumination provides only a weak depth cue for the two overlapping regions inside the circle. Figure (b) shows depth discontinuity curves, where adjacent pixel depths differ by more than a threshold, in blue. The arrows show the average gradients of discontinuity curves. Figure (c) shows the proximity shadows cast by the discontinuity curves in (b).*

spatial relationship immediately becomes clear. Such pairs of visible patches result in a depth discontinuity that usually occurs along one or more silhouette curves as shown in Figure 2.11 (b). We use proximity shadows to show the relative distance between the two overlapping patches if the eye-space distance between them is within a predefined threshold.

To compute proximity shadows, we first identify the depth discontinuity curves by comparing the value of each pixel in the depth map with its neighbors. We then generate a shadow light direction for each depth discontinuity curve by using the depth gradient. The shadow light direction is determined by rotating the direction vector to the viewer by a small angle θ towards the average depth-gradient direction as shown in Figure 2.12. Finally we use the shadow light direction in a shadow map to cast proximity shadow for the depth discontinuity curve. We repeat this for all depth discontinuity curves.

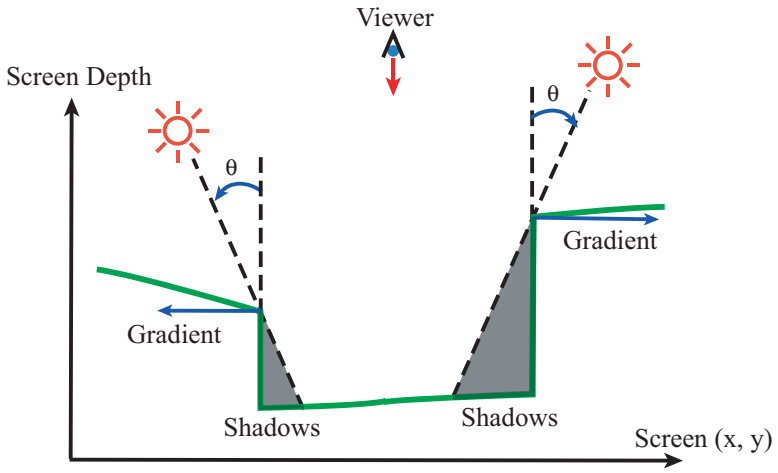
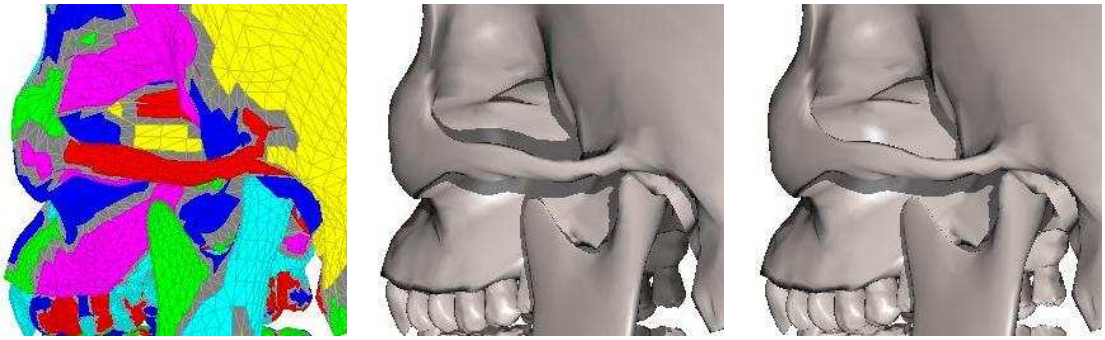


Figure 2.12: *The placement of a light for proximity shadow: At each depth discontinuity curve of the depth map, a light for the proximity shadow is placed by rotating a vector to the viewer by an angle θ along the direction of the local gradient.*

While casting proximity shadows, we have to be aware that a narrow region might cause a problem if it has depth discontinuities on multiple sides. If we cast shadows of this region in each direction, it can produce a somewhat disconcerting effect as shown in Figure 2.13(b). For such situations one can use any heuristic that consistently picks one side of the region over the others. Examples of such heuristics may include picking the side of the discontinuity region that is on the left and the top, or pick the side of the discontinuity region that has more surface points on the discontinuity curve (refer Figure 2.13(c)).

2.6 Efficient Computation by Spherical Harmonics

The light placement and light assignment stages are the most time consuming in our lighting design pipeline. In this section, we discuss how to speed up the overall system by efficiently computing and updating the light placement function using



(a) (b) (c)

Figure 2.13: *Avoiding conflicts in proximity shadows: (a) Depth discontinuity curves arise along both sides of the upper cheek bone. (b) The discontinuity curves result in proximity shadows on both sides of bone that might appear disconcerting. (c) This can be fixed by eliminating one of the two proximity shadows.*

the spherical-harmonic-basis representation. The Light Collages process described in Section 2.4 takes a few hundred seconds for a model with tens of thousands of vertices. It is reasonable running time for one-time image generation, but not fast enough for interactive visualization or generation of a large number of images. Also, we might want to store the precomputed light placement functions for interactive rendering. In that case the current representation will need large amounts of storage.

Spherical harmonics (SH) can encode a function defined over a sphere with orthonormal basis functions. Spherical harmonics can represent any function with representational accuracy related to the number of coefficients used. Since our light placement functions and weight functions are defined on a sphere, we can encode them using spherical harmonics. Moreover, since our light placement function and weight functions are low frequency, we can represent them with a small number of spherical harmonic coefficients resulting in efficient storage and computation.

2.6.1 Spherical Harmonics Background

The spherical-harmonic (SH) basis functions (Figure 2.14) with the parametrization $(x, y, z) = (sin\theta cos\varphi, sin\theta sin\varphi, cos\theta)$ are defined as

$$y_l^m(\theta, \varphi) = \begin{cases} \sqrt{2} K_l^m cos(m\varphi) P_l^m(cos\theta), & m > 0 \\ \sqrt{2} K_l^m sin(-m\varphi) P_l^{-m}(cos\theta), & m < 0 \\ \sqrt{2} K_l^0 P_l^0(cos\theta), & m = 0 \end{cases}$$

where P_l^m are the associated Legendre polynomials and K_l^m are defined as:

$$K_l^m = \sqrt{\frac{(2l+1)(l-|m|)!}{4\pi(l+|m|)!}}$$

We can project a scalar function f defined on a sphere onto its SH coefficients h, through the integral: $h(m, l) = \int f(s) y_l^m(s) ds$, where s represents directions (θ, φ).

We approximate the function f with these coefficients h by using n SH bands:

$$\tilde{f}(s) = \sum_{l=0}^{n-1} \sum_{m=-l}^{l} h(m, l) y_l^m(s)$$

We can usually save space for storing the function f by using spherical harmonics since we can approximate the original function with a small number of SH coefficients depending on the accuracy we need. Figure 2.14 shows the first nine SH basis functions.

Since the SH basis functions are orthonormal, we can compute many operations such as addition, subtraction, and inner product between two functions efficiently

by applying the operations to the SH coefficients of the functions. For example, let h_f and h_g be the SH coefficients of two functions f and g. We can compute the SH coefficients of $f + g$ by simply adding h_f and h_g. Therefore, we can approximate $f + g$ with $h_f + h_g$, i.e. $(\tilde{f} + \tilde{g})(s) = \sum_{l=0}^{n-1} \sum_{m=-l}^{l} (h_f(m, l) + h_g(m, l)) y_l^m(s)$.

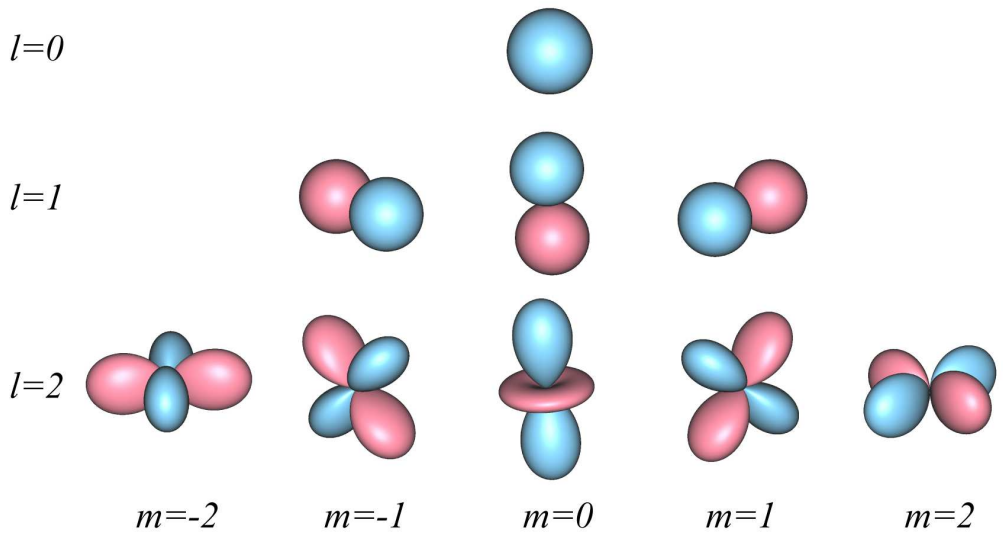


Figure 2.14: *Spherical harmonic basis functions of the first three bands: Absolute values of basis functions are plotted as distances from the center. Positive values are painted in blue, and negative values are painted in red.*

Spherical harmonics are rotationally invariant. This means that if we rotate a spherical-harmonic representation of a function, it will be exactly the same as the spherical harmonic representation of the rotated function. Let R be a rotation operator and $SH(f)$ be the spherical-harmonic representation of a function f. Then we can denote the rotationally-invariant property as: $SH(R(f)) = R(SH(f))$. The following matrix shows the structure of a spherical harmonic rotation matrix.

$$R = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & X & X & X & 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & X & X & X & 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & X & X & X & 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & X & X & X & X & X & \dots \\ 0 & 0 & 0 & 0 & X & X & X & X & X & \dots \\ 0 & 0 & 0 & 0 & X & X & X & X & X & \dots \\ 0 & 0 & 0 & 0 & X & X & X & X & X & \dots \\ 0 & 0 & 0 & 0 & X & X & X & X & X & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

The rotational-invariance property of spherical harmonics enables us to rotate a function by multiplying a rotation matrix to the vector of SH coefficients. There are several methods for computing this matrix by using recurrence relations [9, 14, 42]. We use Blanco *et al.*'s method [9] for fast rotations of spherical harmonic representations. Their method incrementally computes a rotation matrix of real spherical harmonics by using the recursive relations between matrix components for adjacent bands.

We can describe an arbitrary 3D rotation in terms of Euler angles (α, β, γ) rotated in the order of Z, Y, and Z axis. Let $d^l_{m,n}$ be the element at the row m and column n of the rotation matrix R for the spherical-harmonic band l. Then Blanco *et al.* [9] denote R as:

$$R = \begin{bmatrix}
d^0_{0,0} & 0 & 0 & 0 & 0 & 0 & 0 & \dots & & \\
0 & d^1_{-1,-1} & d^1_{-1,0} & d^1_{-1,1} & 0 & 0 & 0 & 0 & 0 & \dots \\
0 & d^1_{0,-1} & d^1_{0,0} & d^1_{0,1} & 0 & 0 & 0 & 0 & 0 & \dots \\
0 & d^1_{1,-1} & d^1_{1,0} & d^1_{1,1} & 0 & 0 & 0 & 0 & 0 & \dots \\
0 & 0 & 0 & 0 & d^2_{-2,-2} & d^2_{-2,-1} & d^2_{-2,0} & d^2_{-2,1} & d^2_{-2,2} & \dots \\
0 & 0 & 0 & 0 & d^2_{-1,-2} & d^2_{-1,-1} & d^2_{-1,0} & d^2_{-1,1} & d^2_{-1,2} & \dots \\
0 & 0 & 0 & 0 & d^2_{0,-2} & d^2_{0,-1} & d^2_{0,0} & d^2_{0,1} & d^2_{0,2} & \dots \\
0 & 0 & 0 & 0 & d^2_{1,-2} & d^2_{1,-1} & d^2_{1,0} & d^2_{1,1} & d^2_{1,2} & \dots \\
0 & 0 & 0 & 0 & d^2_{2,-2} & d^2_{2,-1} & d^2_{2,0} & d^2_{2,1} & d^2_{2,2} & \dots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots
\end{bmatrix}$$

Blanco *et al.*'s method [9] computes the elements of R for the bands 0 and 1 analytically. They then compute the elements for further bands using symmetric properties as well as recursive relations between elements:

$$\begin{aligned}
d^l_{m,m'} &= \frac{l(2l-1)}{\sqrt{(l^2-m^2)(l^2-m'^2)}}\{(d^1_{0,0} - \frac{mm'}{l(l-1)})d^{l-1}_{m,m'} \\
&- \frac{\sqrt{[(l-1)^2-m^2][(l-1)^2-(m')^2]}}{(l-1)(2l-1)}d^{l-2}_{m,m'}\} && (2.1)\\
d^l_{l,l} &= d^1_{1,1}d^{l-1}_{l-1,l-1} && (2.2)\\
d^{l-1}_{l-1,l-1} &= (ld^1_{0,0} - l + 1)d^{l-1}_{l-1,l-1} && (2.3)\\
d^l_{l,m-1} &= -\sqrt{\frac{l+m}{l-m+1}}tan\frac{\beta}{2}d^l_{l,m} && (2.4)\\
d^l_{l-1,m-1} &= -\frac{lcos\beta - m + 1}{lcos\beta - m}\sqrt{\frac{l+m}{l-m+1}}tan\frac{\beta}{2}d^l_{l-1,m} && (2.5)\\
& && (2.6)
\end{aligned}$$

2.6.2 SH-Based Light Placement and Assignment

We can efficiently represent and compute the light placement function by using spherical harmonics. Let $h(l, m)$ be the spherical-harmonic coefficients for representing the light placement function P: $h(l, m) = \int P(s) y_l^m(s) ds$. We approximate the light placement function P with the coefficients as:

$$\tilde{P}(s) = \sum_{l=0}^{n-1} \sum_{m=-l}^{l} h_i(m, l) y_l^m(s)$$

Recall from Section 2.4 that the overall light placement function is a sum of specular and diffuse weight functions from each vertex. We observe that given two vertices i and j with the same curvature but different normals, the weight functions (diffuse and specular) of vertex i can be computed by rotating the corresponding weight functions (diffuse and specular) of vertex j. Therefore, we can pre-compute the spherical-harmonic representations of the weight functions for each curvature value and simply rotate them according to the per-vertex normals. This is significantly more efficient than repeatedly projecting every vertex's specular and diffuse weight functions into spherical-harmonic coefficients.

First, we pre-compute the specular and diffuse weight functions for a canonical normal $\overrightarrow{n_0}$ for each curvature intensity c. We currently sample c uniformly in the range 0 to 1. Let the specular and diffuse weight functions for $\overrightarrow{n_0}$ be represented by spherical-harmonic coefficients $f_0(l, m, c)$ and $g_0(l, m, c)$, respectively:

$$f_0(l,m,c) = \int S(0,s) y_l^m(s) ds$$
$$g_0(l,m,c) = \int D(0,s) y_l^m(s) ds$$

Second, for each vertex i, we find the pre-computed specular and diffuse weight functions whose curvature value is closest to the vertex's curvature value c_i. We rotate these weight functions to get the weight functions of the vertex i. Let $R_{0\to i}$ be the spherical-harmonic rotation matrix that is equivalent to the rotation of $\overrightarrow{n_0}$ to the normal $\overrightarrow{n_i}$ of vertex i. Then we compute the spherical-harmonic coefficients $f_i(l,m)$ and $g_i(l,m)$ as:

$$f_i(l,m) = R_{0\to i} f_0(l,m,c_i)$$

$$g_i(l,m) = R_{0\to i} g_0(l,m,c_i)$$

We compute the spherical-harmonic coefficients of the light placement function by adding the $f_i(l,m)$ and $g_i(l,m)$ for all vertices:

$$h(l,m) = \sum_i (f_i(l,m) + g_i(l,m))$$

In Section 2.4-B, we discussed an iterative scheme for identifying the best light source directions using the light placement function. According to this scheme we

identify a light l_i and assign it to patches p_j best lighted by it. We then deduct from the light placement function, the contributions from the weight functions of all the vertices in the patches p_j lighted by light l_i. We do this directly with the spherical-harmonic coefficients of the light placement and per-vertex weight functions. Since the spherical harmonic functions define an orthonormal basis, we simply subtract the spherical-harmonic coefficients of the per-vertex weight functions from the spherical-harmonic coefficients of the light placement function.

2.7 Minimality of the Light Sources

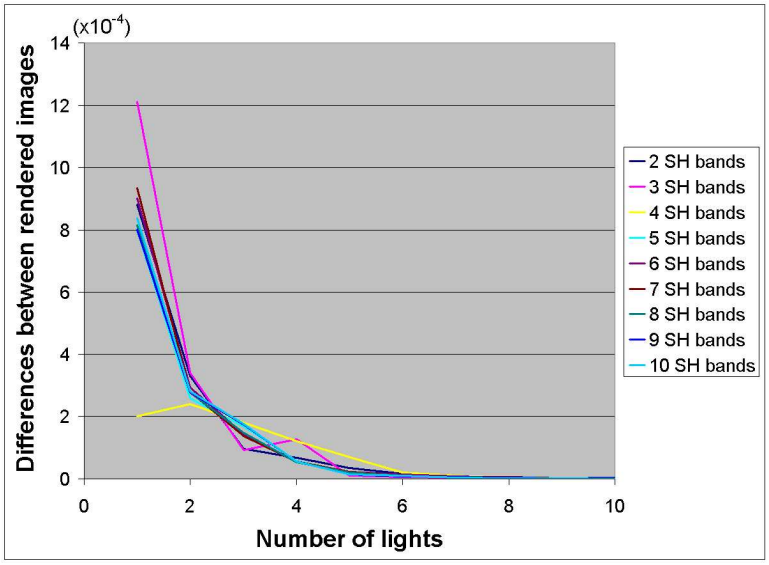


Figure 2.15: *The image differences with different numbers of light sources (Skull Model)*

The choice of an appropriate number of discrepant light sources is important and requires trade-offs between quality and efficiency. If we arbitrarily choose the number of discrepant light sources, our rendered image may be of an undetermined quality for different objects. If we choose too many lights, we will pay for the extra

run-time lighting costs and if we choose too few lights we may not have an adequate number of lights to show the fine geometric detail.

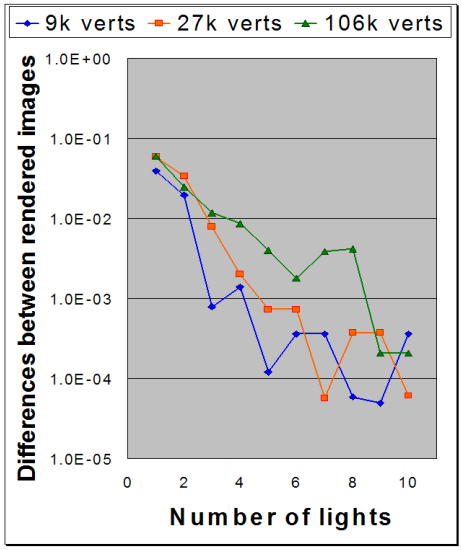
The Light Collages framework selects light sources incrementally. We examine the incremental improvement in the quality of the image by adding an extra light. If the image improvement (measured as the root-mean-square difference) is small enough we can stop adding light sources. In this dissertation, we stop adding light sources when fewer than 2% of the screen pixels change by less than 2% of their color range. For example, if we use a screen with a 1024×768 resolution and 8-bit colors, less than $16K$ pixels are allowed to vary by less than 5 out of 256 color values. In this case, the RMSD threshold works out to be 2.8×10^{-3}. The graph in Figure 2.15 shows that 8 lights suffice for the Skull model. We note that in Figure 2.15 the image differences are nearly independent of the number of spherical-harmonic coefficients. Therefore, efficient computation by using low-band spherical-harmonic representation is quite appropriate for determining the number of light sources.

Discrepant lighting by more lights increases the geometric detail that we can see. This improvement diminishes for a given geometric level of detail after a certain number of lights have been added (see Figure 2.16(d)). This leads us to believe that it should be possible to relate the level of detail for lighting with the level of detail for geometry. Thus, less-detailed lighting should suffice for less-detailed geometry whereas higher-detailed geometry should require higher-detailed lighting. Just as the geometric level-of-detail systems manage the complexity of geometry based on parameters such as the viewer position relative to the object one should manage the

(a) 9K vertices

(b) 27K vertices

(c) 106K vertices



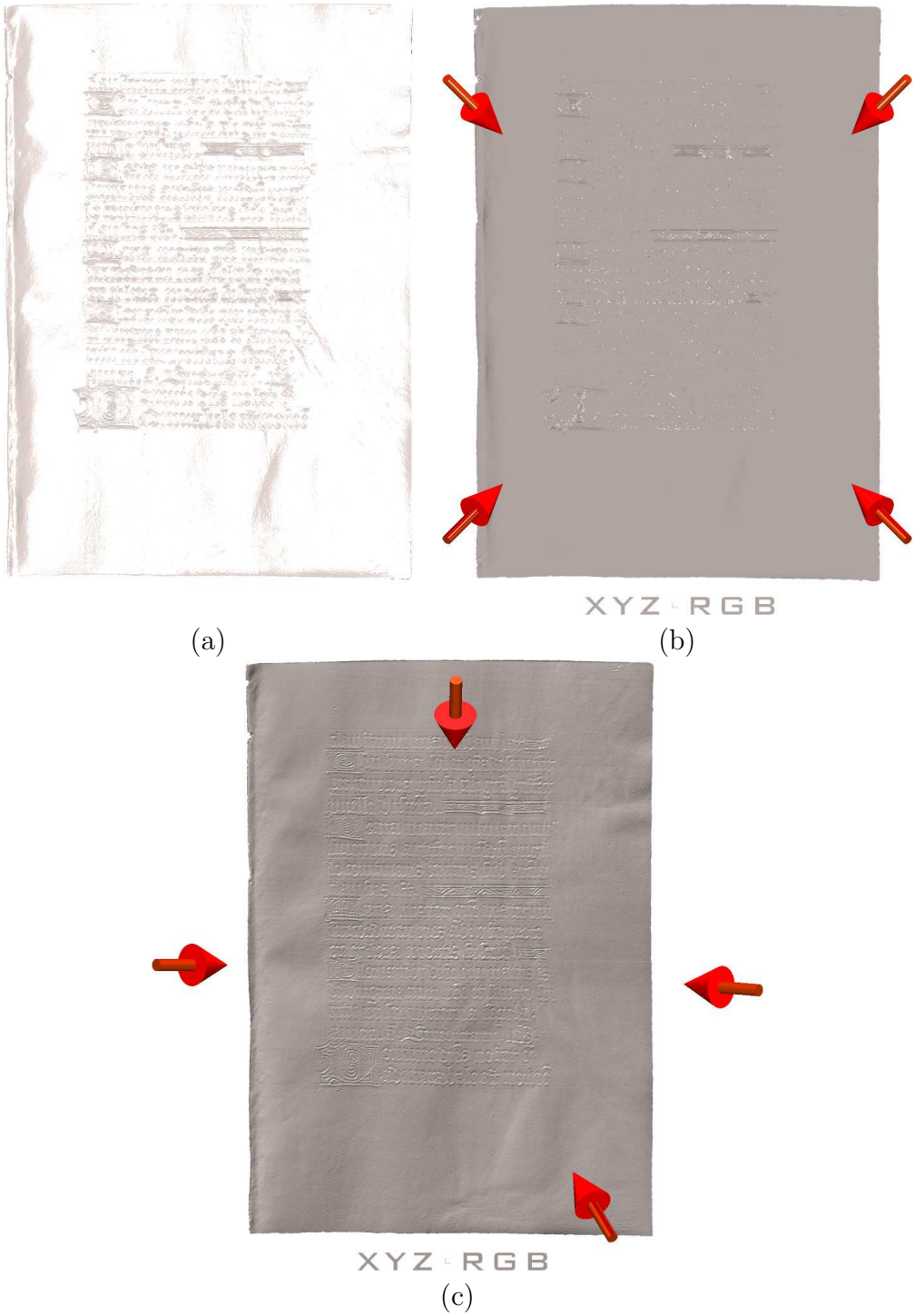
(d)

Figure 2.16: *Minimality of light sources for varying levels of detail in geometry: Images (a), (b), and (c) show Light Collages rendering of Dama De Elche model represented with 9K, 27K, and 106K vertices. Image (d) shows the image differences with different numbers of light sources for each level of detail. In general, a mesh with less detail requires fewer discrepant lights than a mesh with more detail. Meshes with 9K, 27K, and 106K vertices select 2, 3, and 5 lights with our system.*

lighting level of detail based on the geometry and viewing parameters. Figures 2.16 (a), (b), and (c) show Light Collages rendering of the Dama De Elche model at different geometric levels of detail. Figure 2.16(d) shows that a higher level of detail in geometry requires more lights than a less-detailed geometry.

2.8 Results

Figures 2.17 and 2.18 show the visualization results using our system. The manuscript dataset used in Figure 2.17 was provided to us by Paul Debevec at USC and scanned by XYZ RGB Inc. The manuscript is a 177×163mm page from a 15th century "Book of Hours" produced near Rouen in France. The scanned manuscript has an accuracy of $100\mu m$ horizontally and vertically, and $3\mu m$ along the depth. At a depth resolution of $3\mu m$, the scan is detailed enough to lift the impressions of the ink. Naive consistent lighting as shown in Figures 2.17 (a) and (b) fails to capture the fine details of the characters and the subtle variations and wrinkles in the manuscript. Figure 2.17(c) nicely shows these subtle variations in the geometry with our geometry-dependent discrepant lighting. We have used the same lighting models and material properties for generating all the three images. As you can see in (a), the specular highlight from consistent lighting will sometimes cause large bright areas on flat regions, while highlights from our method (c) are only on highly curved regions. This helps elucidate geometry details. In Figure 2.18(a) we show the result from lighting the Pelvis model by consistent lighting with 4 lights, and (b)–(d) show the results by Light Collages. The proximity shadow cast by the sacrum and the



(a) (b)

(c)

Figure 2.17: *Light Collages for the Rouen Manuscript: (a) Rendered by one consistent light placed along the view direction, (b) Rendered by four consistent lights arranged at the front four corners of a cube, and (c) Rendered by Light Collages with four lights using* 25 *SH coefficients.*

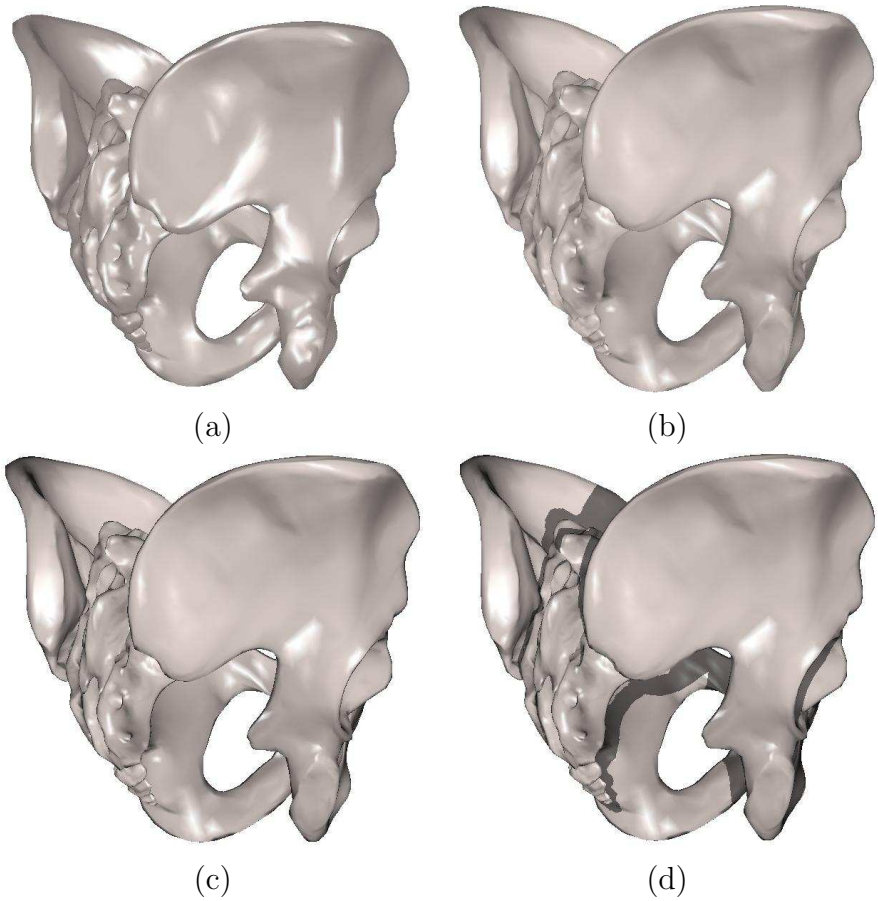
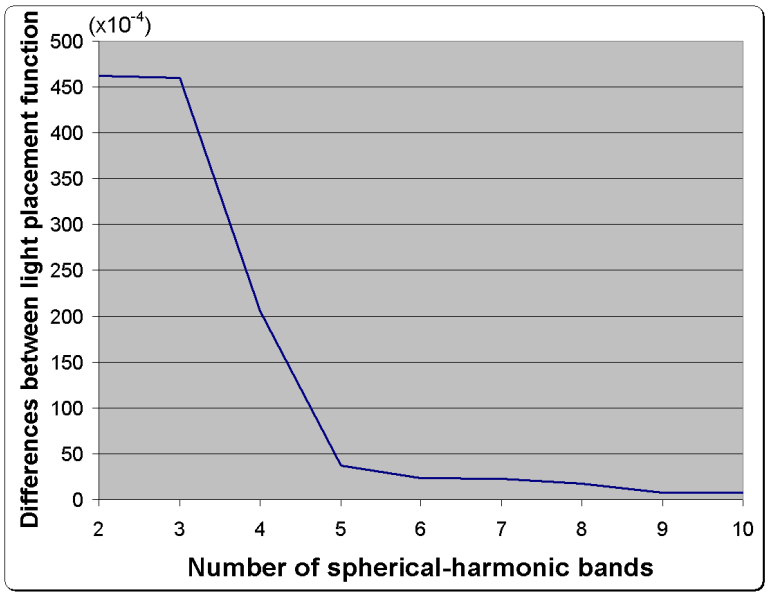


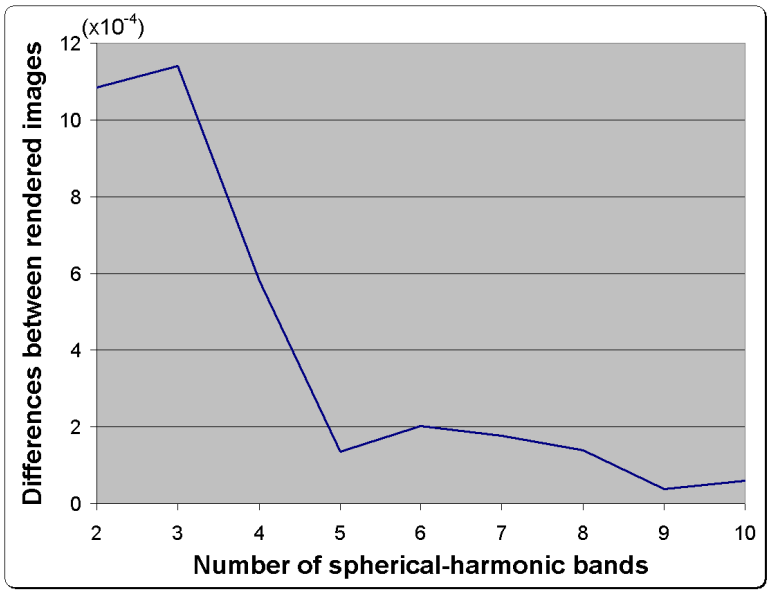
Figure 2.18: *Lighting design for the Pelvis model. We used 25 SH coefficients for generating images (b)–(d). (a) shows consistent rendering with four lights at the front four vertices of a cube, and (b) shows Light Collages rendering by 4 lights. In image (c), we further added silhouette lighting, and in (d) we further added proximity shadows.*

right coxal bone in Figure 2.18(d) nicely illustrates the depth relationship between adjacent regions of the pelvis.

We have reduced the computation time for each vertex to be proportional to the number of spherical-harmonic coefficients instead of the number of directional samples. In this dissertation we have used 12518 (approximately $12K$) directions. The important question that remains to be addressed is how many spherical-harmonic coefficients are necessary to give us an acceptable level of accuracy. To



(a)



(b)

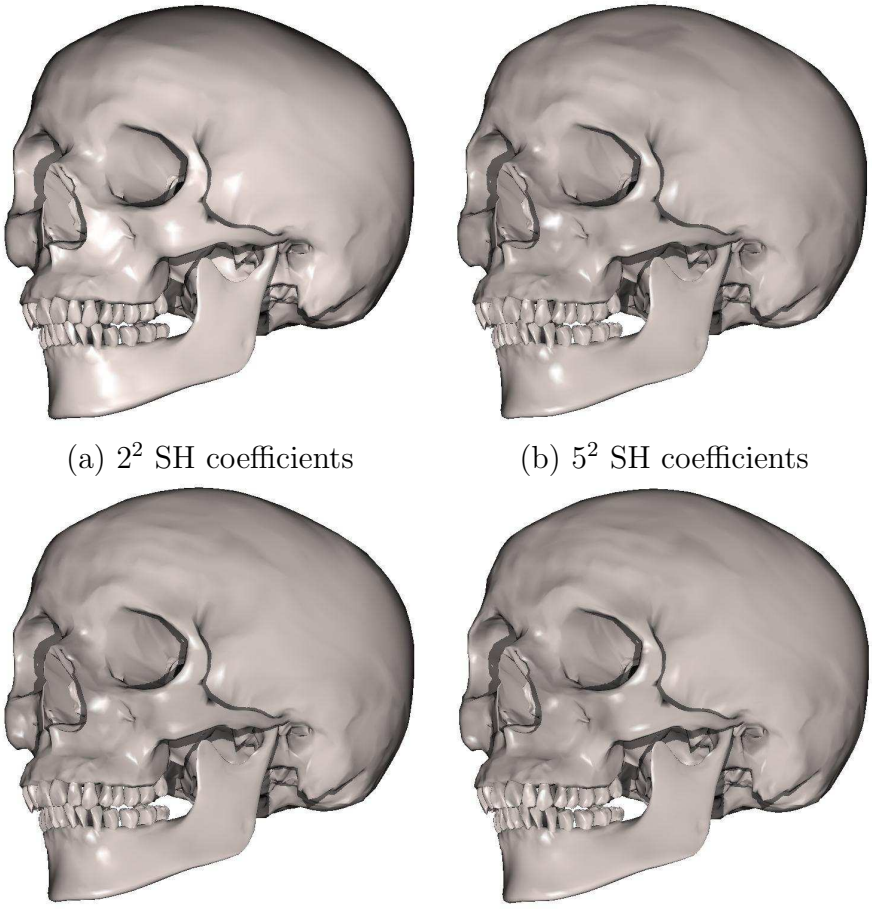
Figure 2.19: *We show the difference from using spherical harmonics compared to direct evaluation over* $12K$ *uniformly distributed light directions for the* $33K$ *vertex Skull model. Figure (a) shows the root-mean-square difference between direct and spherical-harmonic evaluation of the normalized light placement function. Figure (b) shows the root-mean-square difference between images resulting from lighting design with direct computation and with spherical harmonics.*

address this, we compared the accuracy of the lighting design process with and without spherical harmonic representations. To compare the accuracy in representing the light placement function, we first normalized the light placement function to be in the range 0 to 1. Then, for each of the approximately $12K$ light directions we computed the difference between direct evaluation and the spherical-harmonic evaluation, and used these to compute the overall root-mean-squared difference. This is shown in Figure 2.19(a) over an increasing number of spherical-harmonic bands for the Skull model. The number of spherical-harmonic coefficients used is the square of the number of bands used. In Figure 2.19(b) we show the root-mean-square difference between images of the Skull model rendered using lighting design with and without spherical harmonics.

Table 2.1: Run Times for Light Placement and Assignment

Model	**Skull (33K verts)**	**Pelvis (17K verts)**
SH Bands	**Time (sec)**	**Time (sec)**
2	4.04	2.54
3	5.30	3.41
4	8.05	5.19
5	13.42	7.11
6	19.85	12.05
7	29.58	16.62
8	42.81	23.63
9	60.36	35.51
10	81.30	43.72
Direct Computation	234.17	138.82

As you can see in Figures 2.19 and 2.20, the error is reduced significantly when the number of spherical harmonic bands is five or greater. We report the timings for light placement and assignment in Table 2.1. These times are for a Pentium IV, 1.5 GHz system with 1GB RAM. As one can see, the spherical-harmonic method with



(a) 2^2 SH coefficients (b) 5^2 SH coefficients

(c) 8^2 SH coefficients (d) Direct Computation

Figure 2.20: *Lighting for Skull: (a)–(c) show Light Collages rendering with various spherical harmonic coefficients, and (d) shows the result with direct computation.*

5 bands is almost 20 times faster than the direct computation. Further, since we only need to store 25 spherical-harmonic coefficients per vertex instead of over $12K$ directional samples, our spherical-harmonic-based lighting design approach reduces the required memory by a factor of over 500.

2.9 Conclusions

We have introduced the concept of geometry-dependent lighting which allows discrepant lights dependent on local geometry to only affect local regions. Our Light

Collages system uses geometry-dependent lighting for automatic lighting design for effective visualization of scientific datasets. Our method relies on using multiple light sources that can be used for accurate local lighting on surfaces, with possible global inconsistencies.

The human visual system is remarkably adept at inferring shape from largely local cues and recent research [67] suggests that inconsistencies in illumination may not be resolved at a low level. However, it is also believed that the human visual system has a strong preference for a single illumination from above which if violated may lead to incorrect perception of shape [74]. Elder *et al.* [21] reconcile these by suggesting that for simple objects and scenes the low-level human visual system might expect and process consistent illumination but for more complex scenes and objects, with multiple light sources and inter-reflections, discrepancies in illuminations might require higher-level processing. We find it interesting that our results on minimality of the number of light sources derived by our Light Collages system, show an increase in the number of discrepant lights with increasing geometric detail. However, in the absence of an adequate computational model that can reconcile these opposing points of view, it might be desirable to allow a user to modify the light source directions for regions in which a system such as Light Collages, might cause ambiguous or incorrect shape interpretation.

We have shown how our method can incorporate silhouette lighting as well as proximity shadows to further elucidate the local structure of the scientific datasets. We believe our method greatly improves the visualization while retaining the look and feel of traditional 3D graphics rendering. In addition to the visual appear-

ance, interactivity is essential for the perception of 3D shapes. We use spherical-harmonics-based representations to efficiently compute the light placement function for use in a real-time system. Our current running times could be further enhanced by using the vertex shaders on modern graphics processors. We also have presented a method to optimize the number of light sources needed for generating images without loss of image quality. This minimality of the light sources depends on the geometry of various models as well as the geometric level of detail of a single model.

Chapter 3

Mesh Saliency

Research over the last decade has built a solid mathematical foundation for representation and analysis of 3D meshes in graphics and geometric modeling. Much of this work however does not explicitly incorporate models of low-level human visual attention. In this chapter, we introduce the idea of *mesh saliency* as a measure of regional importance for graphics meshes. Our notion of saliency is inspired by low-level human visual system cues. We define mesh saliency in a scale-dependent manner using a center-surround operator on Gaussian-weighted mean curvatures. We observe that such a definition of mesh saliency is able to capture what most would classify as visually interesting regions on a mesh. The human-perception-inspired importance measure computed by our mesh saliency operator results in more visually pleasing results in processing and viewing of 3D meshes, compared to using a purely geometric measure of shape, such as curvature. We discuss how mesh saliency can be incorporated in graphics applications such as mesh simplification and viewpoint selection and present examples that show visually appealing results from using mesh saliency.

3.1 Related Work

Low-level cues influence where in an image people will look and pay attention. Many computational models of this have been proposed. Koch and Ullman's [54] early model suggested that salient image locations will be distinct from their surroundings. Our approach is explicitly based on the model of Itti *et al.* [41]. They combine information from center-surround mechanisms applied to different feature maps, computed at different scales, to compute a *saliency map* that assigns a saliency value to each image pixel. Tsotsos *et al.* [94], Milanese *et al.* [65], Rosenholtz [81], and many others describe other interesting saliency models. Among their many applications, 2D saliency maps have been applied to selectively compress [73] or shrink [13, 91] images. DeCarlo and Santella [17] use saliency determined from a person's eye movements to simplify an image producing a non-photorealistic, painterly rendering.

More recently, saliency algorithms have been applied to views of 3D models. Yee *et al.* [101] use Itti *et al.*'s algorithm to compute a saliency map of a coarsely rendered 2D projection of a 3D dynamic scene. They use this to help decide where to focus computational resources in producing a more accurate rendering. Mantiuk *et al.* [61] use a real-time, 2D saliency algorithm to guide MPEG compression of an animation of a 3D scene. Frintrop *et al.* [24] use a saliency map to speed up the detection of objects in 3D data. They combine saliency maps computed from 2D images representing scene depth and intensities. Howlett [39] demonstrate the potential value of saliency for the simplification of 3D models. Their work captures

saliency by using an eye-tracker to record where a person has looked at a 2D image of a 3D model.

These prior works determine saliency for a 3D model by finding saliency in its 2D projection. There is little work that determines saliency directly from 3D structure. Guy and Medioni [32] proposed a method for computing a saliency map for edges in a 2D image, (such edge-based saliency maps were previously explored by Shashua and Ullman [84]). In [63] they extend this framework to apply to 3D data. However, their approach is mainly designed to smoothly interpolate sparse, noisy 3D data to find surfaces. They do not compute an analog to the saliency map for a 3D object. Watanabe and Belyaev [98] have proposed a method to identify regions in meshes where principal curvatures have locally maximal values along one of the principal directions (typically along ridges and ravines). Hisada *et al.* [37] have proposed a method to detect salient ridges and ravines by computing the 3D skeleton and finding non-manifold points on the skeletal edges and associated surface points.

3.2 Mesh Saliency Computation

Itti *et al.* [41]'s method is one of the most effective techniques for computing saliency for 2D images. Our method for computing saliency for 3D meshes uses their center-surround operation. Unlike images, where color is the most important attribute, we consider geometry of meshes to be the most important contributor to saliency. At present our method for mesh saliency uses only geometry, but it

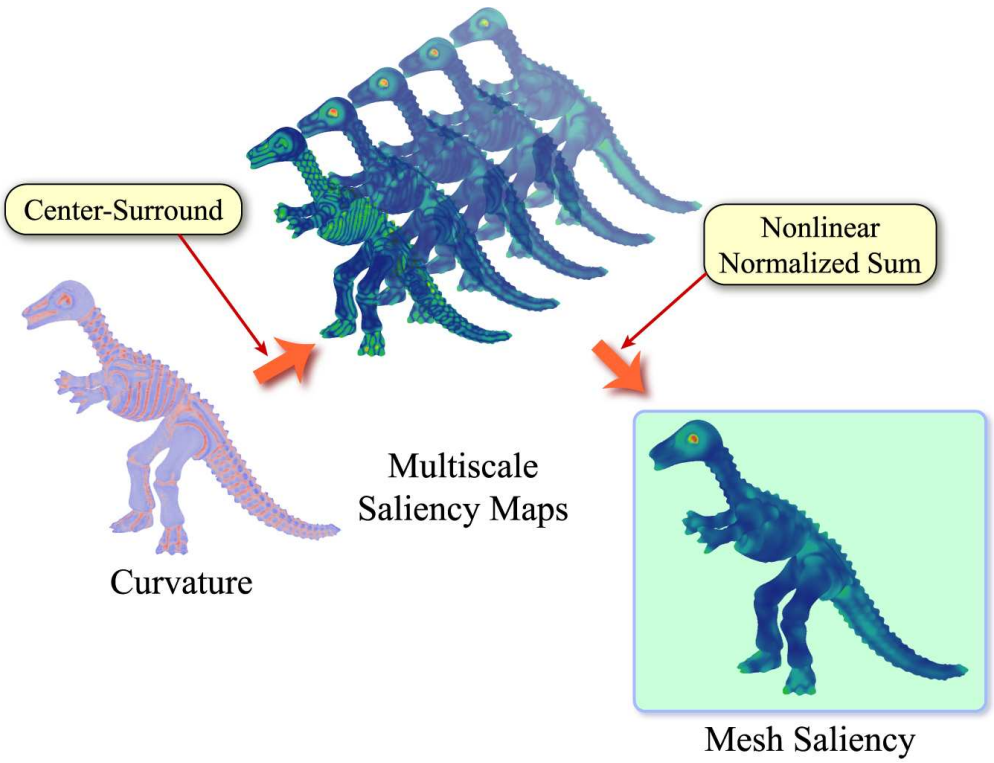


Figure 3.1: *Mesh Saliency Computation: We first compute mean curvature at mesh vertices. For each vertex, saliency is computed as the difference between mean curvatures filtered with a narrow and a broad Gaussian. For each Gaussian, we compute the Gaussian-weighted average of the curvatures of vertices within a radius 2σ, where σ is Gaussian's standard deviation. We compute saliency at different scales by varying σ. The final saliency is the aggregate of the saliency at all scales with a non-linear normalization.*

should be easy to incorporate other surface appearance attributes into it as well. There are several possible characteristics of mesh geometry that could be used for saliency. Before we decide on one let us compare the desiderata of saliency in a 2D image with the saliency of a 3D object. Zero saliency in an image corresponds to a region with uniform intensity. The motivation behind this is that the key image property whose variations are critical is the intensity. In an image, intensity is a function of shape and lighting. For 3D objects however, we have the opportunity to determine the saliency based on shape, independent of lighting. For 3D objects,

we feel that a sphere is the canonical zero-saliency feature. This is in spite of the fact that depending on the lighting, a sphere may not produce a uniform intensity image. In the case of the sphere the property that is invariant is the curvature. Therefore we are guided by the intuition that it is changes in the curvature that lead to saliency or non-saliency. This has led us to formulate mesh saliency in terms of the mean curvature used with the center-surround mechanism. Figure 3.1 gives an overview of our saliency computation.

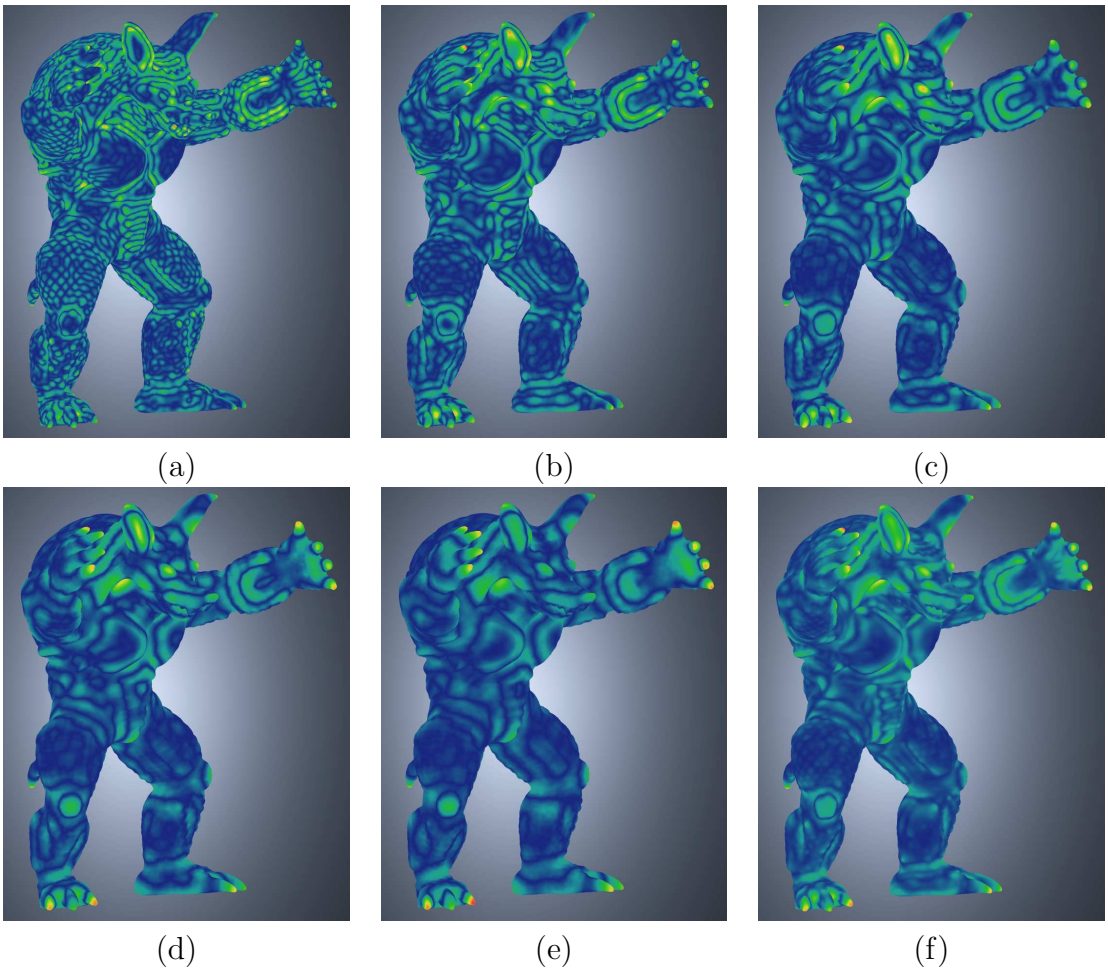


Figure 3.2: *Images (a)–(e) show the saliency at scales of* 2ϵ, 3ϵ, 4ϵ, 5ϵ, *and* 6ϵ. *Image (f) shows the final mesh saliency after aggregating the saliency over multiple scales. Here,* ϵ *is* 0.3% *of the length of the diagonal of the bounding box of the model.*

The first step of our saliency computation involves computing surface curvatures. There are a number of excellent approaches that generalize differential-geometry-based definition of curvatures to discrete meshes [64, 93]. One can use any of these to compute the curvature of a mesh at a vertex v. Let the curvature map $\mathcal{C}$ define a mapping from each vertex of a mesh to its mean curvature, i.e. let $\mathcal{C}(v)$ denote the mean curvature of vertex v. We use Taubin [93]'s method for curvature computation. Let the neighborhood $N(v, \sigma)$ for a vertex v, be the set of points within a distance σ. One can consider several distance functions to define the neighborhood, such as the geodesic or the Euclidean. We have tried both and found that the Euclidean distance gave us better results and that is what we use here. Thus, the neighborhood $N(v, \sigma)$ for a vertex v is $N(v, \sigma) = \{x | \|x - v\| < \sigma,\ \text{x is a mesh point}\}$. Let $G(\mathcal{C}(v), \sigma)$ denote the Gaussian-weighted average of the mean curvature. We compute this as:

$$G(\mathcal{C}(v), \sigma) = \frac{\sum_{x \in N(v, 2\sigma)} \mathcal{C}(x) exp[-\|x - v\|^2/(2\sigma^2)]}{\sum_{x \in N(v, 2\sigma)} exp[-\|x - v\|^2/(2\sigma^2)]}$$

Note that with the above formulation, we are assuming a cut-off for the Gaussian filter at a distance 2σ. We compute the saliency $\mathcal{S}(v)$ of a vertex v as the absolute difference between the Gaussian-weighted averages computed at fine and coarse scales. We currently use the standard deviation for the coarse scale as twice that of the fine scale:

$$\mathcal{S}(v) = |G(\mathcal{C}(v), \sigma) - G(\mathcal{C}(v), 2\sigma)|$$

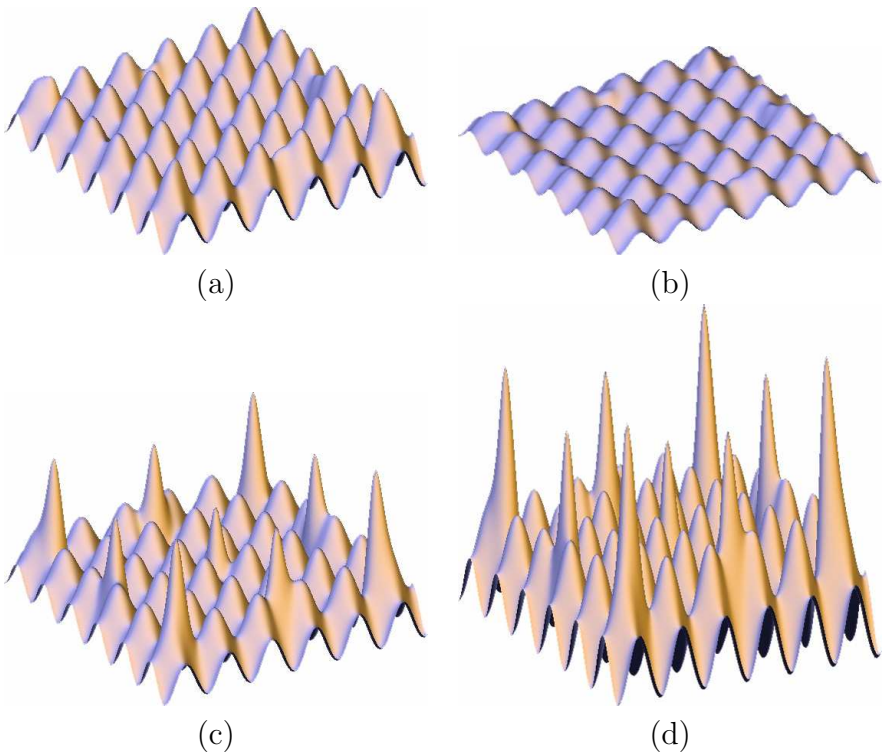


Figure 3.3: *In aggregating the saliency over multiple scales, we use the notion of suppression for emphasizing more informative scales. Imagine a saliency map where many points have high saliency (Figure (a)) and a saliency map where few points have high saliency (Figure (c)). We consider the second saliency map more informative. For each scale, we first normalize the saliency map and then multiply each saliency with the difference between the maximum saliency at that level and the average local maxima. In this example, the saliency for the model in Figure (a) is suppressed by multiplying with small factor resulting in Figure (b) and the saliency for Figure (c) is promoted by multiplying with a larger factor resulting in Figure (d)*

To compute mesh saliency at multiple scales, we define the saliency of a vertex v at a scale level i as $\mathcal{S}_i(v)$:

$$\mathcal{S}_i(v) = |G(\mathcal{C}(v), \sigma_i) - G(\mathcal{C}(v), 2\sigma_i)|$$

where, σ_i is the standard deviation of the Gaussian filter at scale i. For all the results in this dissertation we have used five scales $\sigma_i \in \{2\epsilon, 3\epsilon, 4\epsilon, 5\epsilon, 6\epsilon\}$, where ϵ is 0.3% of the length of the diagonal of the bounding box of the model.

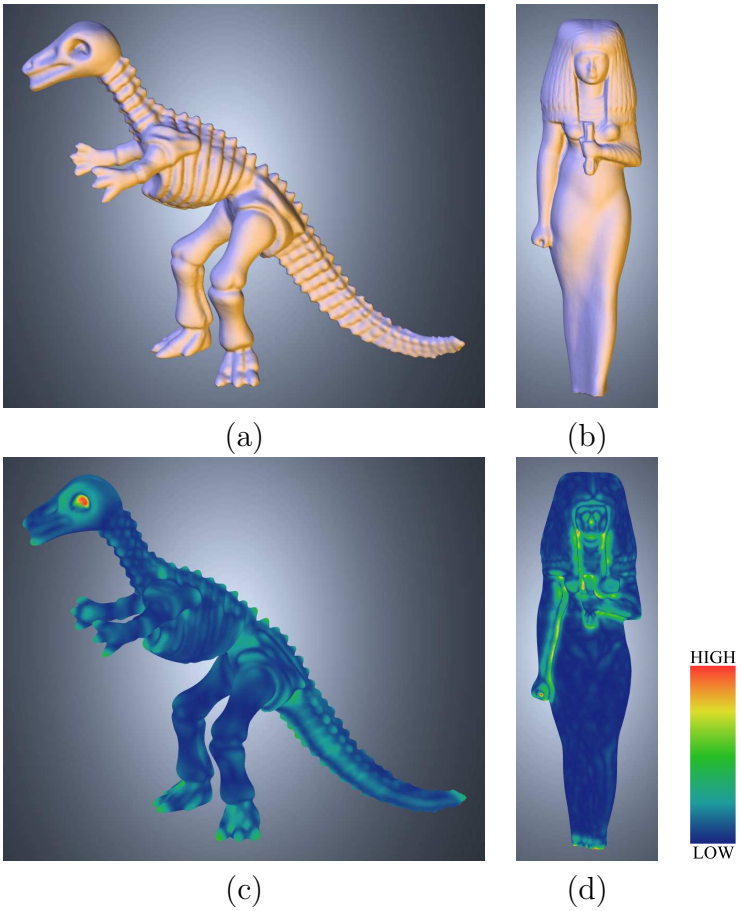


Figure 3.4: *We show mesh saliency for the Cyberware Dinosaur model (a) in figure (c) and for the Cyberware Isis model (b) in figure (d). Warmer colors (reds and yellows) show high saliency and cooler colors (greens and blues) show low saliency.*

For combining saliency maps $\mathcal{S}_i$ at different scales, we apply a non-linear suppression operator S similar to the one proposed by Itti *et al.* [41]. This suppression operator promotes saliency maps with a small number of high peaks (Figure 3.2(e) or Figure 3.3 (c)) while suppressing saliency maps with a large number of similar peaks (Figure 3.2(a) or Figure 3.3 (a)). Thus, non-linear suppression helps us in reducing the number of salient points. If we do not use suppression, we get far too many regions being flagged as salient. We believe, therefore, that this suppression helps to define what makes something unique, and therefore potentially salient. For

each saliency map $\mathcal{S}_i$, we first normalize $\mathcal{S}_i$. We then compute the maximum saliency value M_i and the average $\bar{m}_i$ of the local maxima excluding the global maximum at that scale. Finally, we multiply $\mathcal{S}_i$ by the factor $(M_i - \bar{m}_i)^2$. The final mesh saliency $\mathcal{S}$ is computed by adding the saliency maps at all scales after applying the non-linear normalization of suppression:

$$\mathcal{S} = \sum_i S(\mathcal{S}_i)$$

3.3 Salient Simplification

Mesh simplification is useful in a number of graphics applications including rendering acceleration with a level-of-detail-based rendering scheme. In such a scheme the level of simplification used to render an object is determined by the tradeoff between the desired frame rate and the desired visual fidelity. Thus, a less important object in a scene is rendered using a more simplified mesh and a high-importance object is rendered with little or no simplification. Time-critical applications that require a constant target frame rate often have a fixed triangle budget for each frame. For such applications, achieving a higher level of perceptual fidelity with the same triangle budget is highly desirable.

There is a large and growing body of literature on simplification of meshes using a diverse set of error metrics and simplification operators [59]. Several simplification approaches use estimates of mesh curvature to guide the simplification process and achieve high geometric fidelity for a given triangle budget [52, 96].

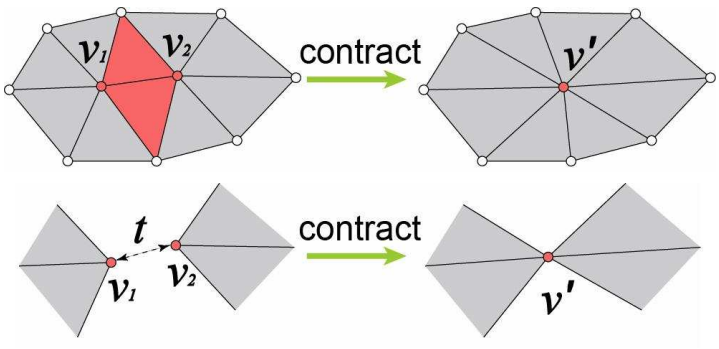
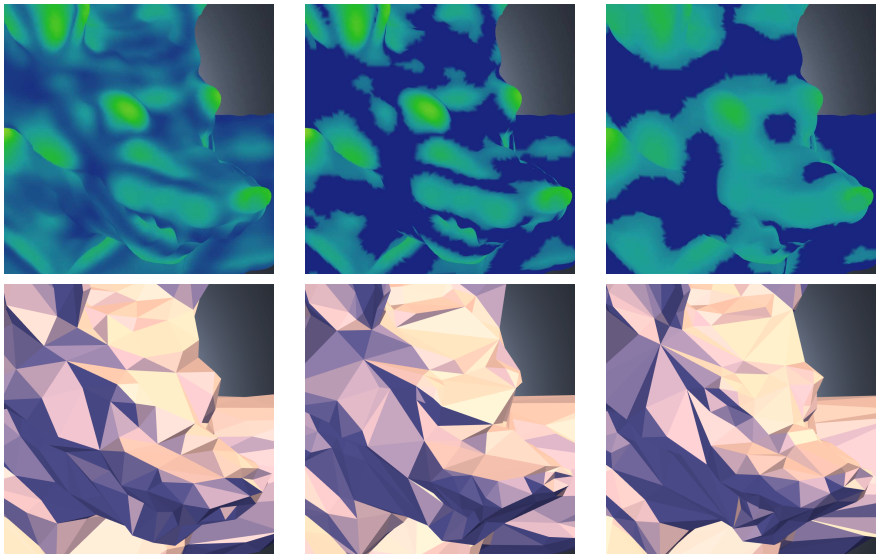


Figure 3.5: *Edge contractions during a mesh simplification process.*

Other simplification approaches, such as QSlim [25], use error metrics that while not directly computing curvature, are related to curvature [36]. Curvature has also been directly used to identify salient regions on meshes. Watanabe and Belyaev [98] classify extrema of the principal curvatures as salient features and preserve them better during simplification. Their method however, does not use a center-surround mechanism to identify regions on a mesh that are different from their local context.

For evaluating the effectiveness of our mesh saliency method, we have modified the quadrics-based simplification method (Qslim) of Garland and Heckbert [25] by weighting the quadrics with mesh saliency. However, it should be equally easy to integrate our mesh saliency with any other mesh simplification scheme. Garland and Heckbert's method simplifies a mesh by repeatedly contracting vertex pairs ordered by increasing quadric errors. Figure 3.5 shows the contraction of vertex pairs. Let P be the set of planes of triangles incident at a vertex v, where the plane $p \in P$ defined by the equation $ax + by + cz + d = 0$, $a^2 + b^2 + c^2 = 1$, is represented as $(a\ b\ c\ d)^T$. Then the quadric for the plane p is defined as $Q_p = pp^T$. They define the error of v with respect to p as the squared distance of v to p which is computed by $v^T Q_p v$. The quadric Q of v is the sum of all the quadrics of neighboring planes: $Q = \sum_{p \in P} Q_p$.



(a) (b) (c)

Figure 3.6: *We show the saliency-based weights and the quality of the* 99% *simplification (*3.5K *triangles) for the Stanford Armadillo model for three choices of the simplification weights: (a) the original mesh saliency ($\mathcal{W} = \mathcal{S}$) (b) the amplified mesh saliency ($\mathcal{W} = A\mathcal{S}$), and (c) the smoothed and amplified mesh saliency ($\mathcal{W} = A(G(\mathcal{S}, 3\epsilon))$).*

After computing quadrics of all vertices, they compute the optimal contraction point $\bar{v}$ for each pair (v_i, v_j) which minimizes the quadric error $\bar{v}^T(Q_i+Q_j)\bar{v}$ where Q_i and Q_j are quadrics of v_i and v_j, respectively. The algorithm iteratively contracts the pair with the minimum contraction cost $\bar{v}^T(Q_i + Q_j)\bar{v}$. After a pair is contracted, the quadric for the new point $\bar{v}$ is computed simply by adding the two quadrics $Q_i + Q_j$.

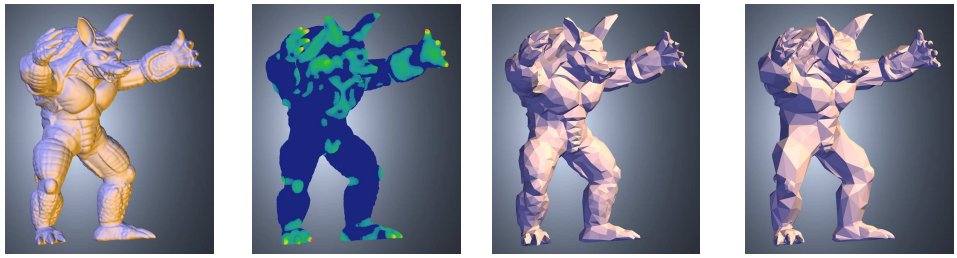
We guide the order of simplification contractions using a weight map $\mathcal{W}$ derived from the mesh saliency map $\mathcal{S}$. We have found that using the simplification weights based on a non-linear amplification of the saliency gives us good results. We believe that the reason behind this is that by amplifying the high saliency vertices we are ensuring that they are preserved longer than the non-salient vertices with high

contraction costs. Specifically, we define a saliency amplification operator A using a threshold α and an amplifying parameter λ, such that we amplify the saliency values that are greater than or equal to α by a factor λ. Thus, the simplification weight map $\mathcal{W}$ using the saliency amplification operator A is specified as:

$$\mathcal{W}(v) = A(\mathcal{S}(v), \alpha, \lambda) = \begin{cases} \lambda\mathcal{S}(v) & \text{if } \mathcal{S}(v) >= \alpha \\ \mathcal{S}(v) & \text{if } \mathcal{S}(v) < \alpha \end{cases}$$

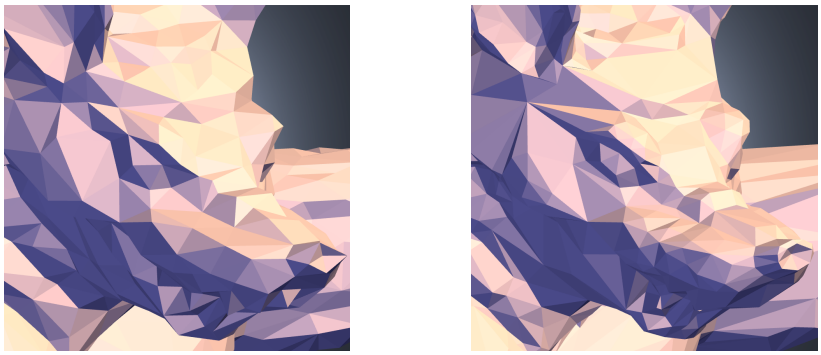
For all the saliency-based simplification results in this dissertation, we use $\lambda = 100$ and $\alpha = 30^{th}$ percentile saliency. At the initialization stage of computing the quadric Q for each vertex v, we multiply Q by its simplification weight $\mathcal{W}(v)$ derived from the saliency of v: $Q \leftarrow \mathcal{W}(v)Q$. Analogous to the computation of a quadric after a vertex-pair collapse, the simplification weight $\mathcal{W}(v)$ for the new vertex v is the sum of the weights for the pair of vertices being collapsed $\mathcal{W}(v_i) + \mathcal{W}(v_j)$.

Obviously, the quality of simplification increases when we apply the saliency amplifying operator. However, we have observed that even when we directly use the saliency as the weighting factor without the amplifying operator, i.e. with $\lambda = 1$, the interesting features are preserved longer than with the original quadric-based method. We have also observed that blurring the saliency map before computing the amplified saliency gives us fewer salient regions and allows the simplification process to focus more on these selected regions. We use $\sigma = 3\epsilon$ for blurring, i.e.$\mathcal{W} = A(G(\mathcal{S}, 3\epsilon))$, This is shown in Figure 3.6. We compute the saliency map just once and do not modify it during simplification so that we can always stay true to the original model's saliency.

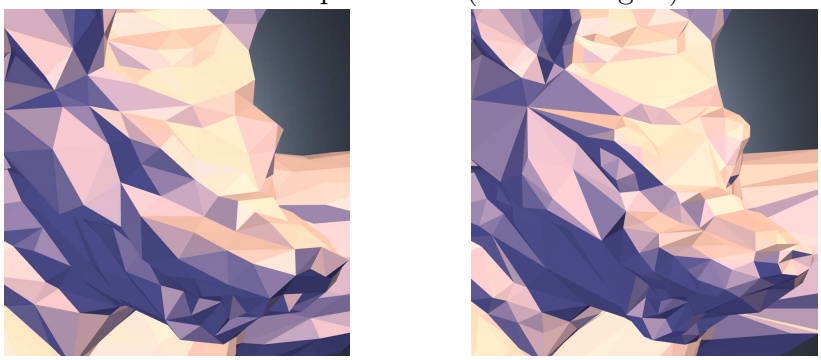


Original (346K triangles) | Saliency | Qslim | Saliency-Based

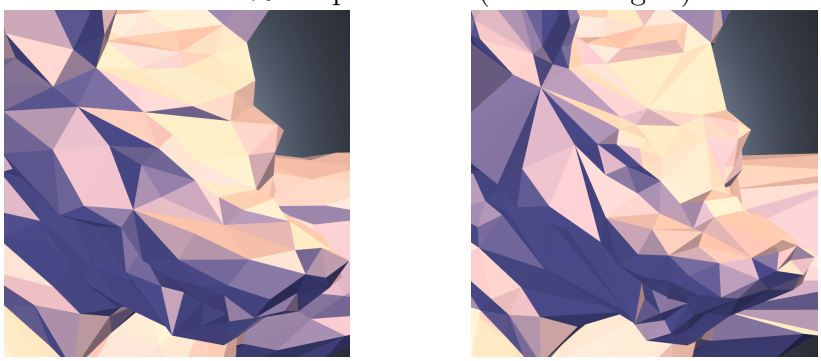
99% simplification (3.5K triangles)



98% simplification (6.9K triangles)



98.5% simplification (5.2K triangles)

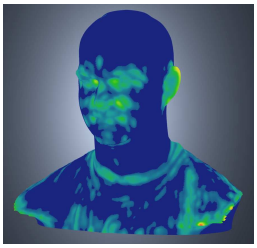


99% simplification (3.5K triangles)

(a) Simplification by Qslim (b) Simplification guided by saliency

Figure 3.7: *Simplification results for the Stanford Armadillo: (a) shows simplified models using Qslim and (b) shows different levels of simplification using saliency. The three right columns show the zoomed-in face of the Armadillo. The eyes and the nose are preserved better with our method while the bumps on the legs are smoothed faster.*

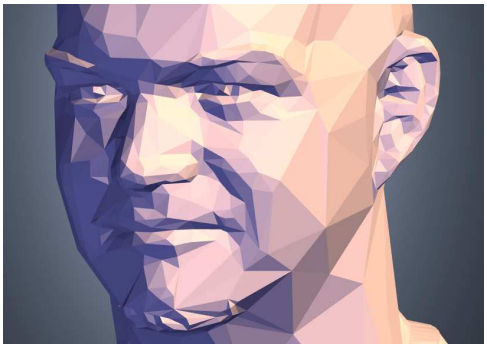
Original
(606K tris)



Saliency

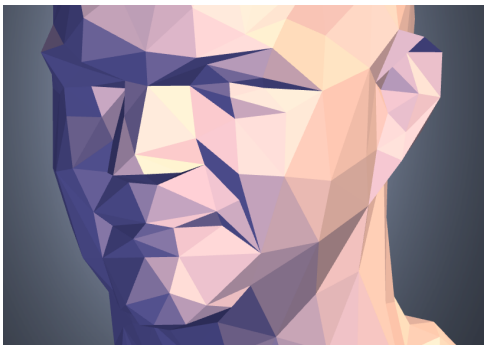
Simplification
by Qslim
4K tris (Smooth Shading)

Simplification
guided by saliency



4K triangles

2K triangles



1K triangles

(a) Simplification by Qslim (b) Simplification guided by saliency

Figure 3.8: *Simplification results for the Cyberware Male: (a) shows simplifications by Garland and Heckbert's method, and (b) shows simplifications by our method using saliency. The eyes, nose, ears, and mouth are preserved better with our method.*

3.4 Salient Viewpoint Selection

With advances in 3D model acquisition technologies, databases of 3D models are evolving to very large collections. Accordingly, the importance of automatically crafting *best* views that maximally elucidate the most important features of an object has also grown for high-quality representative first views, or sequence of views. A number of papers have addressed the problem of automatically selecting a viewpoint for looking at an object. Kamada and Kawai [46] describe a method for selecting views in which surfaces are imaged non-obliquely relative to their normals, using parallel projection. Stoev and Straßer [89] consider a different approach that is more suitable to viewing terrains, in which most surface normals in the scene are similar, and visible scene depth should be maximized. In the context of computer vision, Weinshall and Werman [100] show an equivalence between the most stable and most likely view of an object, and show that this is the view in which an object is flattest. Finding the optimal set of views of an object for purposes of image-based rendering has also been considered, using measures such as those providing best coverage of the scene [23], and those that provide the most information [97].

Blanz *et al.* [10] have conducted user studies to determine the factors that influence the preferred views for 3D objects. They conclude that selection of a preferred view is a result of complex interactions between task, object geometry, and object familiarity. Their studies support visibility (and occlusion) of salient features of an object as one of the factors influencing the selection of a preferred view. Gooch *et al.* [30] have built a system that uses art-inspired principles and some of the factors

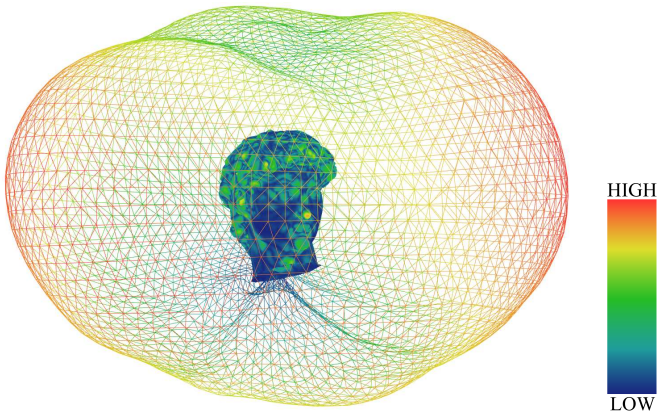


Figure 3.9: *For viewpoint selection, we find the viewpoint that maximizes the visible saliency sum. Here, the wireframe mesh around the David's head model shows the magnitude of the visible saliency sum when the model is seen from each direction. The color of the mesh is also mapped from the visible saliency sum. Our method selects the view-direction with the highest magnitude.*

suggested by Blanz *et al.* [10] to automatically compute initial viewpoints for 3D objects. Systems such as these can greatly benefit from a computational model of mesh saliency.

We have developed a method for automatically selecting viewpoint so as to visualize the most salient object features. Our method selects the viewpoint that maximizes the sum of the saliency for visible regions of the object. For a given viewpoint v, let $F(v)$ be the set of surface points visible from v, and let $\mathcal{S}$ be the mesh saliency. We compute the saliency visible from v as: $U(v) = \sum_{x \in F(v)} \mathcal{S}(x)$. Then the viewpoint with maximum visible saliency v_m is defined as $v_m = \underset{v}{argmax}\ U(v)$. One possible solution here is to exhaustively compute the maximum visible saliency over all viewpoints. This is shown in Figure 3.9. This, however, could get computationally intensive as the amount and complexity of 3D content rises.

Instead, we use a gradient-descent-based optimization heuristic to help us select good viewpoints. The optimization variables are the longitude and latitude,

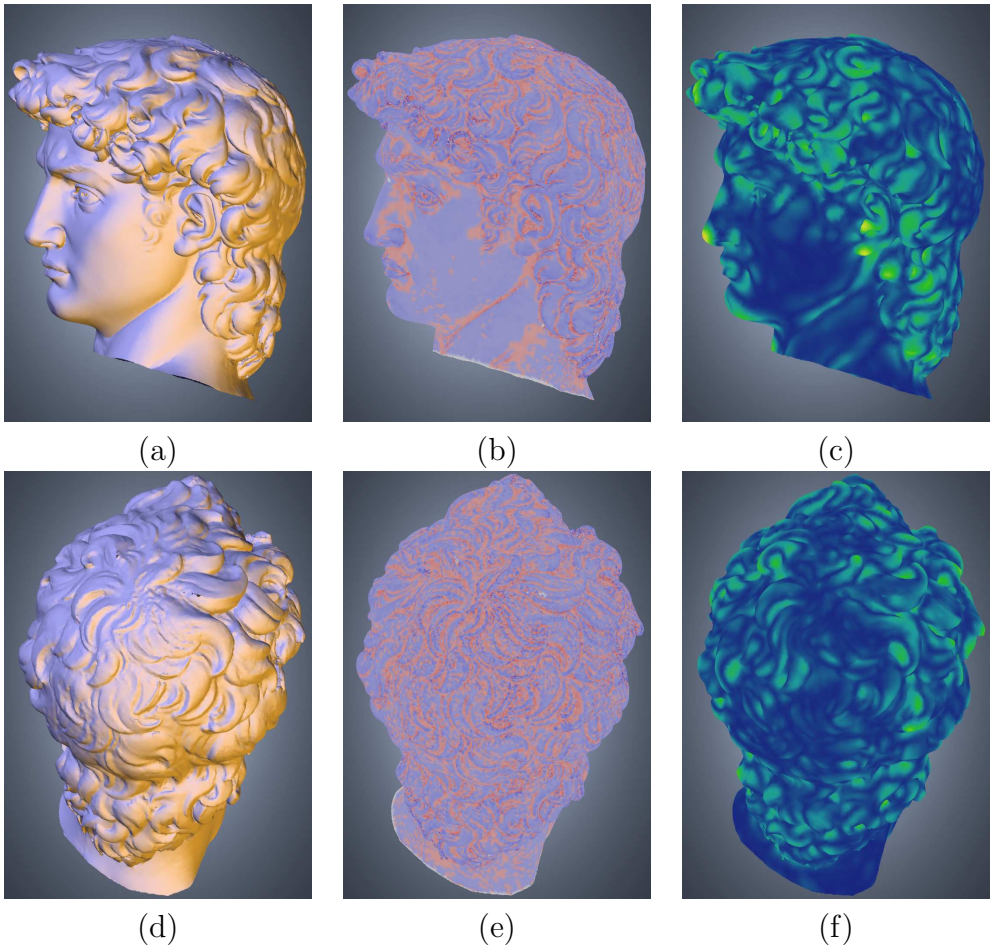


Figure 3.10: *Image (a) shows a viewpoint selected by maximizing visible saliency, and image (d) shows a viewpoint selected by maximizing visible mean curvature. Images (b) and (e) show the mean curvature for the two selected viewpoints, and images (c) and (f) show the saliency. Since saliency negates the repeated hair texture in image (e), the method based on saliency selects the more interesting region of face instead of the top of the head.*

(θ, ϕ) and the objective function is the visible saliency $U(\theta, \phi)$. We start from a random view direction and use the iterative gradient-descent method to find the local maxima. We compute the local gradient by probing the saliency at neighboring view points. We use a randomized algorithm to find the global maximum by repeating this procedure with multiple randomly selected starting points. We can see the results of this approach for Stanford's David model in Figure 3.10. It is interesting

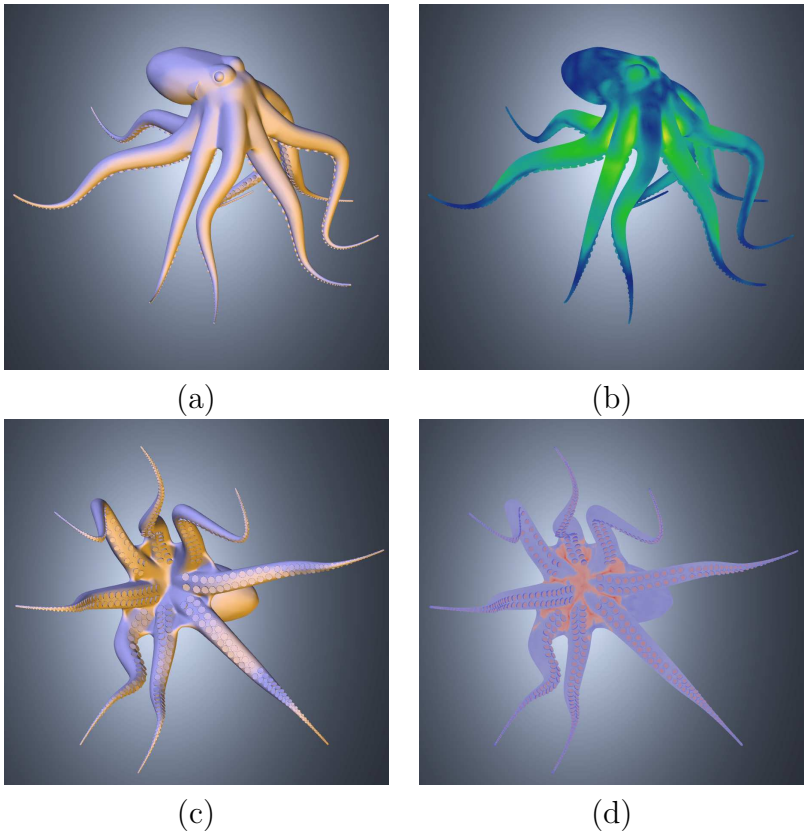


Figure 3.11: *Viewpoint selection for the Octopus model. Images (a)–(b) show the viewpoint selected by maximizing visible saliency, and images (c)–(d) show the viewpoint selected by maximizing visible mean curvature. Image (b) shows the saliency, and image (d) shows the mean curvature. Compared with a curvature-based viewpoint selection method, the saliency-based method picks a more pleasing view for models with repeated textures such as the octopus.*

to see that our approach identified a side of the face whereas a purely curvature-based approach has identified a view looking straight down at the back of David's head.

3.5 Results and Discussion

We have developed a model for mesh saliency, discussed its computation, and shown its applicability to mesh simplification and viewpoint selection. Figure 3.4 shows the mesh saliency for the Cyberware Dinosaur and the Cyberware Isis models.

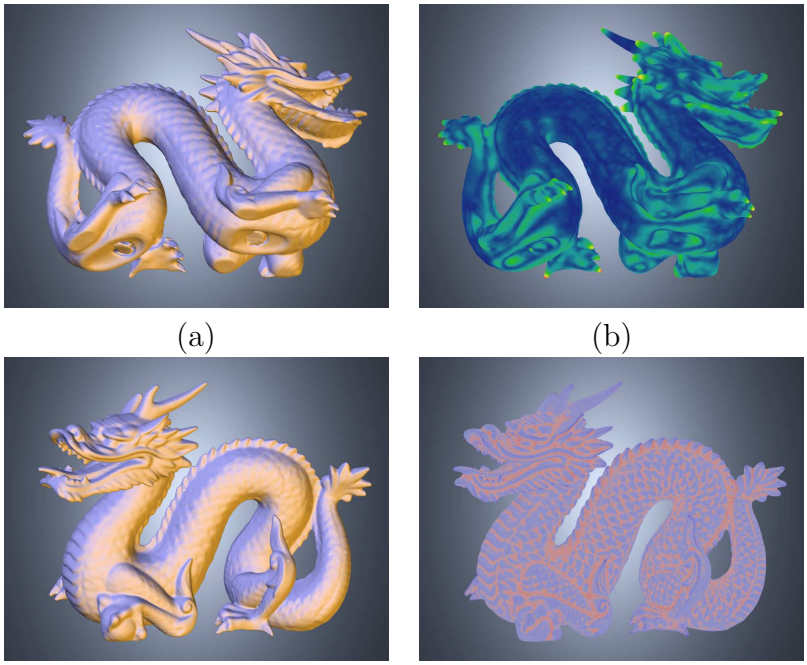


Figure 3.12: *Viewpoint selection for the Stanford Dragon model. Images (a)–(b) show the viewpoint selected by maximizing visible saliency, and images (c)–(d) show the viewpoint selected by maximizing visible mean curvature. Image (b) shows the saliency, and image (d) shows the mean curvature. In this example, we can not say that the saliency-based method picks a more pleasing view compared with a curvature-based viewpoint selection method even though the Dragon model has repeated textures. Our method for saliency-guided view selection for the Dragon selects the view from below instead of from the side since the Dragon's feet have a very high saliency.*

Repeating patterns are usually not classified as salient by our approach. Notice that although the curvature of the Dinosaur's ribs in Figure 3.4 is high, their saliency is low. For other examples, consider the repeated bumps on the legs of the Armadillo model in Figure 3.7, David's hair in Figure 3.10, or patterns in Isis's wig in Figure 3.4. Our approach assigns a low saliency to such local repeating patterns.

The application of our saliency models to guide simplification of meshes have also given us very effective results. Consider for instance the Cyberware Male in Figure 3.8. Notice how our saliency-based simplification retains more triangles around

the ears, nose, lips, and eyes than previous methods. Although in this case, salient simplification preserves the desirable high curvature regions, it can also selectively ignore the undesirable high curvature regions, such as in the simplification of the Armadillo's legs (Figure 3.7) or in ignoring David's hair for viewpoint selection (Figure 3.10).

Figures 3.10–3.12 show the viewpoints selected by our saliency-based method compared with the viewpoints selected by a curvature-based method. In Figures 3.10–3.11, the saliency-based method picks more pleasing views compared while a curvature-based method prioritize views focusing on repeated textures. Figure 3.12 shows where the view selected by our method could disagree with an aesthetic view. The view selected by a curvature-based method is more pleasing, but our method picks the view from below since Dragon's feet have a high saliency. This involves a high-level semantic cue that a face is more interesting than feet. However, our method considers only low-level cues where feet and the holes at the bottom are as salient as the face. In this case, the view selected by our method might contain more geometric information than the view selected by a curvature-based method.

example, we can not say that the saliency-based method picks a more pleasing view compared with a curvature-based viewpoint selection method even though the Dragon model has repeated textures. Our method for saliency-guided view selection for the Dragon selects the view from below instead of from the side since the Dragon's feet have a very high saliency.

The time to compute saliency depends on the scale at which it is computed. Larger scales require identification and processing of a larger number of neighbor-

hood vertices and therefore are more time consuming. Spatial data-structures such as a grid or an octree can greatly improve the running time for establishing the neighborhood at a given scale. Table 3.1 shows the time for saliency computation on a 3.0 GHz Pentium IV PC with 2 GB RAM using a regular grid.

Table 3.1: Run Times for Computing Mesh Saliency

		Time for each scale (sec)				
Model	**#verts**	2ϵ	3ϵ	4ϵ	5ϵ	6ϵ
Dinosaur	56K	1.6	3.4	4.8	6.7	9.0
Armadillo	172K	7.6	15.4	20.5	29.8	41.1
Male	303K	20.7	35.2	50.6	71.2	95.2
Dragon	437K	34.8	72.8	93.8	131.9	178.9
David's Head	2M	593.7	1097.2	1407.4	1968.6	2619.7

Our mesh saliency computation approach is based on a center-surround operator, which is present in many models of human vision. We use this approach primarily because it is a straightforward way of finding regions that are unique relative to their surroundings. For this reason, it is plausible that mesh saliency may capture the regions of 3D models that humans will also find salient. Our experiments provide preliminary indications that this may be true.

3.6 Conclusions

We have developed a model of mesh saliency using center-surround filters with Gaussian-weighted curvatures. We have shown how incorporating mesh saliency can visually enhance the results of several graphics tasks such as mesh simplification and viewpoint selection. For a number of examples we have shown in this chapter, one can see that our model of saliency is able to capture what most of us would classify as

interesting regions in meshes. Not all such regions necessarily have high curvature. While we do not claim that our saliency measure is superior to mesh curvature in all respects, we believe that mesh saliency is a good start in merging perceptual criteria inspired by low-level human visual system cues with mathematical measures based on discrete differential geometry for graphics meshes.

Chapter 4

Salient Lighting

Comprehensibility of large and complex 3D models can be greatly enhanced by guiding viewer's attention to important regions. Lighting is crucial to our perception of shape. Careful use of lighting has been widely used in art, scientific illustration, and computer graphics to guide visual attention. In this chapter, we explore how the saliency of 3D objects can be used to guide lighting to emphasize important regions and suppress less important ones.

4.1 Introduction and Related Work

Recent advances in 3D model acquisition technologies have resulted in creation of gigantic meshes with millions to billions of points. The visual overload caused by the dramatically rising complexity of 3D models prevents us from effectively analyzing and understanding their visual depiction. Research in perceptual psychology has shown that a two-component framework controls where the human visual attention is deployed in visual scenes [40, 68]. The bottom-up mechanism guides the viewer's attention to stimulus-based salient regions. The contrast of colors, intensities, and orientations affects the salience for the bottom-up mechanism. The top-down visual attention involves semantic context for the scenes. Without any task-based knowledge for the scenes, viewers observe a scene in a bottom-up way. Naive ren-

dering of complex models often produces a large amount of visual stimuli, especially when used with feature-enhancing lighting design techniques [31, 55, 56, 83]. This defocuses visual attention leading to a less effective understanding of the scene. Rensink *et al.* [80] have found that observers have difficulty in identifying changes in a scene unless their attention is drawn to the changed regions with verbal cues. Their experiments show that (a) the amount of visual information often exceeds our ability to perceive it effectively and (b) guiding visual attention helps to better analyze and understand the scene.

Braun [11] has found that directing the visual attention based on low-level visual cues as well as object saliency plays a significant role in visual search tasks. His experiments involved showing users several objects with different saliencies based on their size, contrast, and patterns. The users' task was to find a target object among them. He found that a high-contrast object surrounded by low-contrast objects is more noticeable than a low-contrast object surrounded by high-contrast objects. His results show that enhancing the contrast of regions helps us perceive them better.

Several researchers have used contrast enhancement for enhancing important features in graphics data. Gooch *et al.* [29] have proposed a method for converting color images to grayscale images while preserving salient features. They maintain the luminance contrasts in a grayscale image to convey the differences in the CIE color space. Rasche *et al.* [76] have also presented a method for removing color while preserving image details by enhancing their contrast in grayscale images. Ehricke *et al.* [20] have developed an algorithm for visualizing vasculature from volume data

by enhancing the contrast of line-like structures detected by pattern recognition techniques. Researchers have also used lighting to enhance contrast. Raskar *et al.* [77] have used multiple flashes to generate non-photorealistic images from photographs by detecting edges. They enhance contrast around silhouettes by using different flashlight directions for improved edge detection. Kachar [43] has proposed a method for improving the illumination of transparent objects in microscope images by enhancing contrast with an oblique light direction.

High contrast can sometimes lower the overall image quality. When the contrast in an image is too high, the regions with low luminance are not readily noticeable. Tone reproduction, or tone mapping, is a mapping from the real-world luminance intensities to the display luminance intensities [6, 19, 22, 69, 79, 95]. Tone mapping is important because the dynamic range of current display devices is much lower than the real-world dynamic range. Therefore, a bright spot in an image such as a bright sunlight coming from the window may make other regions appear too dark. In this case, suppressing contrast between bright regions and dark regions improves overall comprehensibility.

We have not come across any prior research in automatically guiding visual attention in a 3D rendering by varying the illumination contrast across regions. In this chapter, we introduce *salient lighting* to emphasize salient regions by varying the illumination contrast range based on the computed saliency of the surface point. The observations of Cavanagh [12] suggest that we are insensitive to the global inconsistencies of illumination if the local lighting is consistent. The mesh saliency approach described in Chapter 3 can vary rapidly across an object. Applying that

idea directly in a salient lighting approach may cause distracting lighting inconsistencies. To avoid this problem, we use a smoothed version of mesh saliency. We have also experimented with a new saliency-computation method and use it for salient lighting for molecular visualization.

4.2 Salient Lighting Overview

In this section we discuss emphasizing salient regions based on their importance. This involves two steps. First, we define the regional importance, and second we determine how to emphasize salient regions.

In this chapter, we use the computational mesh saliency approach as described in Chapter 3 as an estimate of the regional importance. We will like to note that for some applications where saliency is based on subjective semantics, it might be more appropriate to acquire it through direct user input. Research on lighting design has used interfaces to sketch the desired rendering result directly on the 3D models [71, 72, 82]. Recent techniques for drawing on the 3D meshes [8, 45, 50] could also facilitate building a system for acquiring the regional importance of 3D meshes through user interfaces.

There are many ways of emphasizing salient regions. As we discussed in Section 4.1, one can improve the comprehensibility by enhancing the contrast. We can apply this idea to improve the visualization of salient regions by enhancing the contrast in salient regions and suppressing it in non-salient regions. We can manage the contrast level of the illumination by suitably modifying ambient, diffuse, and

specular lights, as well as material properties.

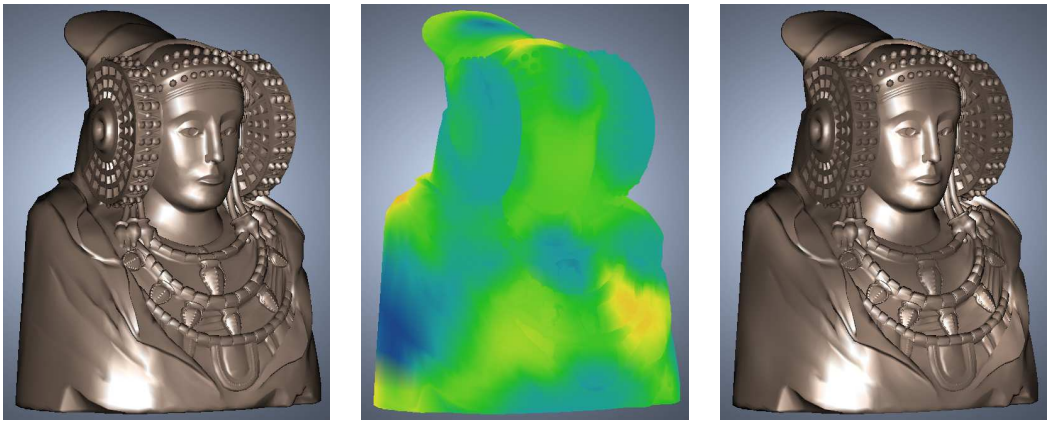
Another method for enhancing the contrast is to change colors based on the saliency. Gooch *et al.* [28] have proposed a non-photorealistic lighting model. They have used warm to cool tones in color transition along the surface-normal changes. This technique helps emphasize the depth cue since we perceive the regions with cool colors to be receded and warm-colored regions to be advanced. We could use this concept to accentuate salient regions in warm colors and suppress non-salient regions in cool colors.

In this chapter, we experiment with a simple approach to show how we can emphasize salient regions by accentuating or attenuating the illumination including ambient, diffuse, and specular lights based on the computed saliency.

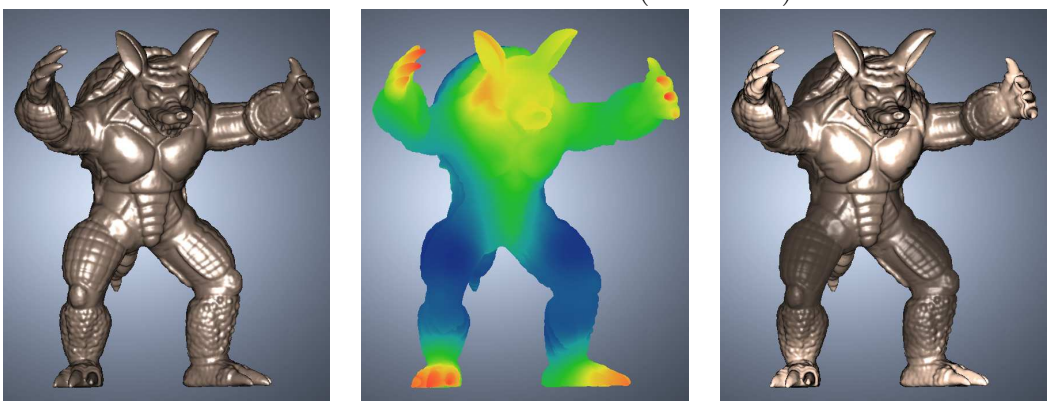
4.3 Curvature-Based Salient Lighting

We have extended the traditional illumination model to merge saliency with lighting to suppress unnecessary or repetitive detail in 3D renderings. Our preliminary results appear in Figure 4.1 (c) where as you can see the distracting specular highlights of the head gear of the Dama De Elche model or the bumps on the leg of the Armadillo model have been largely suppressed. Figure 4.1 (a) shows a rendering with traditional illumination model and (b) shows the saliency of the Dama De Elche and the Armadillo models computed as described below.

Recall from Section 3.2 that we define the saliency of a vertex v at a scale level i as $\mathcal{S}_i(v)$: $\mathcal{S}_i(v) = |G(\mathcal{C}(v), \sigma_i) - G(\mathcal{C}(v), 2\sigma_i)|$, where σ_i is the standard deviation of



The Dama De Elche model (27K verts)



The Armadillo model (172K verts)

(a) Original model (b) Saliency map (c) Salient Lighting

Figure 4.1: *Salient lighting on the Dama De Elche and the Armadillo models.*

the Gaussian filter at scale i and $G(\mathcal{C}(v), \sigma)$ denote the Gaussian-weighted average of the mean curvature with σ as its standard deviation.

Using globally discrepant lighting works fine when the local lighting is consistent. Since rapid changes of saliency between salient and non-salient regions may cause a discontinuity in the illumination, we would like to have a smoothly changing saliency by applying a blurring operator. For making the illumination consistent across reasonably large local regions, we use larger scales than the operator used in Chapter 3. Specifically, we have used five scales $\sigma_i \in \{5\epsilon, 6\epsilon, 7\epsilon, 8\epsilon, 9\epsilon\}$ for generating the result in Figure 4.1, where ϵ is 0.3% of the length of the diago-

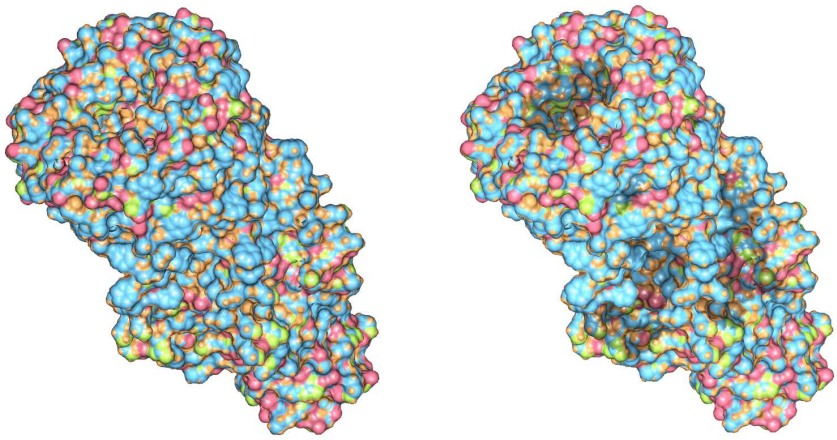
nal of the bounding box of the model. Therefore, the nonlinearly aggregated mesh saliency $\mathcal{S} = \sum_{i=5}^{9} S(\mathcal{S}_i)$. We have also applied the saliency amplification operator A as described in Section 3.3. Recall that given a saliency amplification operator $A(\mathcal{S}(v), \alpha, \lambda)$ with a threshold α and an amplifying parameter λ, we amplify the saliency values that are greater than or equal to α by a factor λ. For the result in Figure 4.1, we use $\lambda = 2$ and $\alpha = 50^{th}$ percentile saliency. We have also observed that blurring the saliency map before and after computing the amplified saliency gives us more smooth changes of the illumination and produces more pleasing results. We use $\sigma = 10\epsilon$ for blurring, i.e. the final saliency map $\mathcal{S}_f$ is computed as: $\mathcal{S}_f = G(A(G(\mathcal{S}, 10\epsilon)), 10\epsilon)$. We define a *salient weight map* $\mathcal{W}$ as a weighting factor for emphasizing or suppressing regions using illumination. Let s_{min} and s_{max} be the minimum and the maximum values of the final saliency map. We compute a salient weight $\mathcal{W}(v)$ at a vertex v by mapping the final saliency value $\mathcal{S}_f(v)$ into a range $[a, b]$:

$$\mathcal{W}(v) = a + \frac{(\mathcal{S}_f(v) - s_{min})(b - a)}{s_{max} - s_{min}}$$

We scale the illumination I_v at v by weight $\mathcal{W}(v)$ while preserving the average illumination intensity. Let $\bar{I}_v$ be the average of conventional illumination over the vertices neighboring v. The region of the neighborhood for a vertex is determined by the largest scale used in saliency computation (which in our current example is 9ϵ). Then we define the salient illumination as: $\mathcal{W}(v)(I_v - \bar{I}_v) + \bar{I}_v$. We use values $a = -0.2$ and $b = 2.5$ for the Dama De Elche model, $a = 0.0$ and $b = 4.0$ for

the Armadillo model. Automatically selecting these values would be an interesting future research topic.

4.4 Normal-Based Salient Lighting

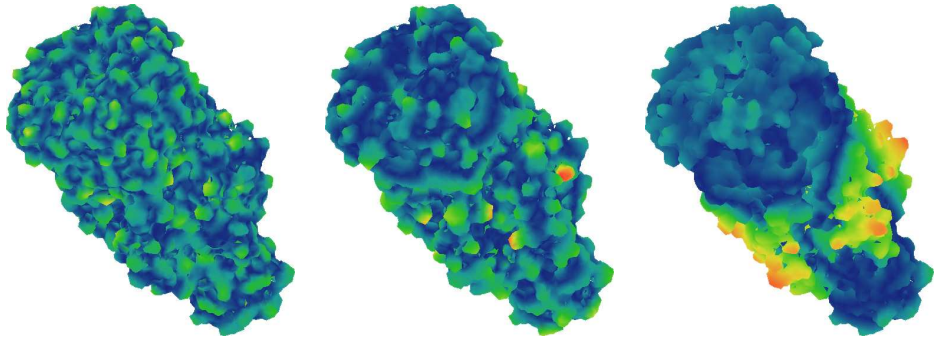


(a) Conventional Illumination (b) Saliency-guided Illumination

Figure 4.2: *Rendering of Ecoli membrane channel. Our saliency-guided rendering clearly shows the central channel as well as the oblique clefts on the side which are not readily apparent with traditional local illumination.*

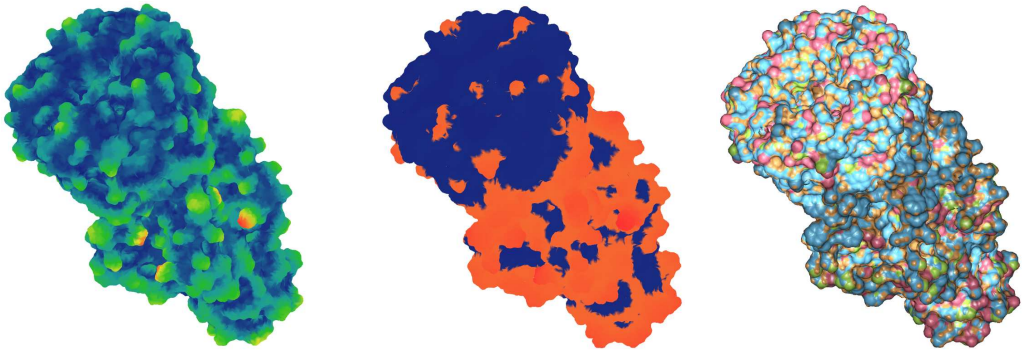
One of the interesting applications for salient lighting is molecular graphics. We are interested in modeling and visualization of protein ion channels that regulate the flow of ions into and out of the cells. The visualization of macromolecular surfaces is challenging since they have rough surfaces due to the spherical representation of atoms. As seen in Figure 4.2 (a), visualization of these surfaces under the traditional local illumination does not adequately reveal the overall structure of the molecule. Using the mesh saliency computed by the center-surround mechanism with mean curvature as in the previous section is also inadequate. For the Ecoli membrane channel in Figure 4.3, the curvature-based mesh saliency starts to show good results from the scale $8\mathring{A}$ and picks the oblique clefts on the side as salient,

but fails to detect the central channel as salient. Figure 4.4 shows that the salient lighting with the curvature-based saliency does not reveal the central channel.



(a) $\sigma_0 = 2\mathring{A}$ (b) $\sigma_1 = 4\mathring{A}$ (c) $\sigma_2 = 8\mathring{A}$

Figure 4.3: *Figures show the saliency maps on the Ecoli membrane channel at multiple scales σ_i.*

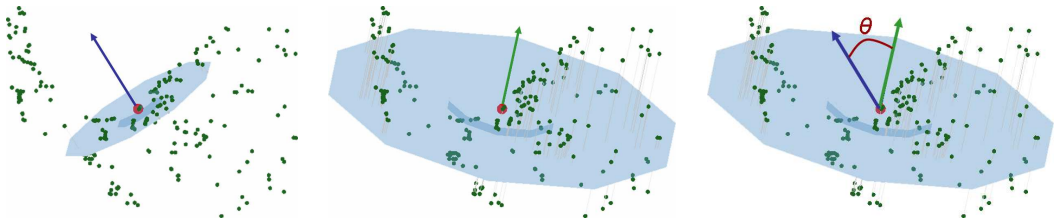


(a)Curvature-based saliency (b) Amplified saliency (c) Salient Lighting

Figure 4.4: *Salient lighting on Ecoli membrane channel based on the saliency computed by applying a center-surround operator to the mean curvature.*

For the vertices of the molecular surface along the walls of the central channel, the mean curvatures at small and large scales are largely similar but the average normal vectors are not. Therefore, the normal-based method is able to better detect the central channel and the oblique clefts on the side as salient. Thus, we apply the center-surround mechanism to normal vectors instead of mean curvature to saliently illuminate molecular surfaces. In this section, we define the salient region

as the region where the distribution of its normal vectors are different from its surroundings.



(a) (b) (c)

Figure 4.5: *Figure (a) shows the plane fitted to the neighboring points at a fine scale and its normal vector, (b) shows the fitting plane and its normal vectors at a coarse scale, and (c) shows the angle between the two normal vectors.*

To compute the difference between normal distributions, we fit planes to two point clouds of the neighborhood at fine and large scales using Principal Component Analysis (PCA) [18] and compute the angle between the normal vectors of the two planes. Figure 4.5 illustrates this operator. Let the normal map $\mathcal{M}$ define a mapping from each vertex of a mesh to its normal vector, i.e. let $\mathcal{M}(v)$ denote the normal vector of the vertex v. Let $V(\mathcal{M}(v), \sigma)$ denote the normal vector of the plane fitted to the points that are within a distance σ from v. We define the saliency of a vertex v at a scale level i as $\mathcal{S}_i(v)$:

$$\mathcal{S}_i(v) = |acos(V(\mathcal{M}(v), \sigma_i) \cdot V(\mathcal{M}(v), 2\sigma_i))|$$

where, σ_i is the distance for generating neighboring points at scale i. We have used hierarchical scales $\sigma_i \in \{2\mathring{A}, 4\mathring{A}, 8\mathring{A}\}$ for this example, and the nonlinearly aggregated mesh saliency $\mathcal{S} = \sum_{i=0}^{2} S(\mathcal{S}_i)$. We have also used the saliency amplification operator $A(\mathcal{S}(v), \alpha, \lambda)$ as described in Section 3.3. We use $\lambda = 10$ and

$\alpha = 60^{th}$ percentile saliency. We use $\sigma = 8\mathring{A}$ for blurring, i.e. the final saliency map $\mathcal{S}_f = A(G(\mathcal{S}, 8\mathring{A}))$.

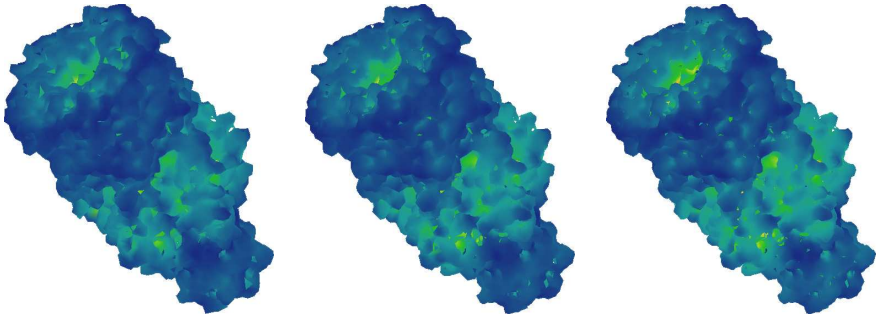
As in Section 4.3, we define a *salient weight map* $\mathcal{W}$ as a weighting factor for emphasizing salient regions. Let s_{min} and s_{max} be the minimum and the maximum values of the final saliency map. We compute a salient weight $\mathcal{W}(v)$ at a vertex v by mapping the final saliency value $\mathcal{S}_f(v)$ into a range $[a, b]$: $\mathcal{W}(v) = a + (\mathcal{S}_f(v) - s_{min})(b - a)/(s_{max} - s_{min})$. For the examples in this section, we use values $a = 0$ and $b = 0.3$.

In this section, we use a constant ambient light and emphasize salient regions by darkening them, i.e. we multiply $1 - \mathcal{W}$ to the diffuse and specular illumination for computing salient illumination.

We have also used multi-resolution techniques for efficiency. We simplified the molecular surface to 5%, 10%, and 20% of the initial number of triangles. We use Qslim [25] for simplification and keep track of the edge contractions so that we know each vertex in the original mesh is contracted to which vertex in the simplified mesh. Then we compute the saliency in the simplified mesh. The saliency in the original mesh is computed by mapping the saliency of contracted vertices to the saliency of original vertices.

4.5 Results and Conclusion

Table 4.1 shows the times for computing saliency of Ecoli membrane channel at different levels of detail, and Figure 4.6 shows the saliency maps. It takes 2 hours 54



(a) $13K$ (b) $26K$ (c) $51K$

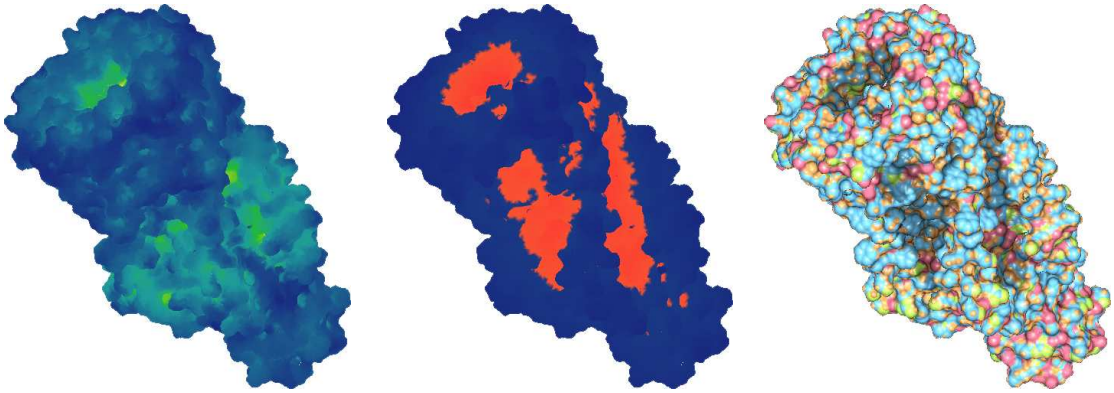
Figure 4.6: *Figures show the saliency maps on the Ecoli membrane channel at multiple levels of detail.*

minutes 18 seconds for computing saliency of the original model with $234K$ vertices at the scale $4\mathring{A}$, while the simplified version with $26K$ vertices takes 37.23 seconds. This gives us a factor of more than 280 speed-up by using a low-resolution mesh with barely any difference as seen in Figure 4.6. Figure 4.7 (a) shows the original saliency without amplification, (b) shows the amplified saliency, and (c) shows the salient lighting with the saliency map shown in (b). As you can see, the salient lighting can clearly illuminate the central channel and the oblique clefts on the side while the traditional local illumination cannot (Figure 4.2 (a)).

Table 4.1: Run Times for Computing Saliency of Ecoli Membrane Channel.

	Time for each scale (sec)		
#Verts	$2\mathring{A}$	$4\mathring{A}$	$8\mathring{A}$
13K	2.00	10.34	49.67
26K	6.54	37.23	206.06
51K	24.51	146.61	929.71

In this chapter, we have proposed a saliency-guided modification of the lighting pipeline to emphasize salient regions. Our approach varies the illumination of a vertex based on its saliency. Salient regions are emphasized with a salient light which is darker or brighter than the lighting on non-salient regions. We have also



(a) Normal-based saliency (b) Amplified saliency (c) Salient lighting

Figure 4.7: *Salient lighting on Ecoli membrane channel based on the saliency computed by applying a center-surround operator to the normal vectors.*

extended the model of mesh saliency described in Chapter 3 for emphasizing salient regions with saliency-guided illumination. We have used center-surround operators with PCA-analyzed normal vectors for dealing with bumpy surfaces in molecular graphics. Our salient lighting can illustrate the global geometric structures such as the central channel and the side oblique clefts in the Ecoli membrane channel.

Chapter 5

Future Work

In this chapter, we present several possible directions for future research. We discuss possible extensions of the research addressed in this dissertation as well as directions for applying the core idea of our research into other domains.

5.1 Lighting Design for Interactive Visualization

Interactivity is essential for the perception of 3D shapes. Smets and Overbeeke [86] have run experiments to complete a jigsaw-like 3D puzzle at different levels of interactivity. They concluded that interactivity is crucial for spatial task performance. Hawkes *et al.* [35] have also insisted that the frame rate is important in enhancing the performance of a virtual reality system. We use spherical-harmonics-based representations to efficiently compute the light placement function for use in a real-time system. Our current running times for Light Collages could be further improved by using the vertex shaders on modern GPUs for building an interactive system. Programmable GPUs provide great flexibility for efficient execution of auxiliary computations in addition to the conventional rendering pipeline. The most time-consuming operations needed at each frame are the rotations, additions, and subtractions of spherical harmonic coefficients at every surface point. Since the surface points can be mapped into a two-dimensional array, these operations can

be computed efficiently with the SPMD structure of GPUs. For more interactivity, the final light assignment information could also be pre-computed and stored for uniformly-sampled view directions. At run time, the light assignment can be determined by interpolating the light positions for the sampled view directions around the current view direction.

Another issue is the smooth transition of the illumination when users change the viewing direction. Our current system designs the lighting based on the current viewing direction. Even though there might be a correlation between the lighting results for similar views, the visual continuity during the interactive movement is not guaranteed. One possible solution is to pre-compute and store the lighting design parameters for several sampled view directions and apply a smooth filter to interpolate between them. We can store the lighting parameters in spherical harmonics for saving space as well as getting a smooth filtering effect. Similar techniques could be used for rendering animations or time-varying data. Another possible solution for animations or time-varying data is to consider four-dimensional input for designing the lighting instead of computing the lights for each frame. We can get continuous lighting changes by restricting the lighting difference along the time dimension. Moreover, we can efficiently compute lighting parameters by considering the time coherence of the geometry.

5.2 Generalized Lighting Design

In our lighting design system we have assumed that the lights are directional. Generalizing our approach to positional point light sources or area light sources would be an interesting direction for future work.

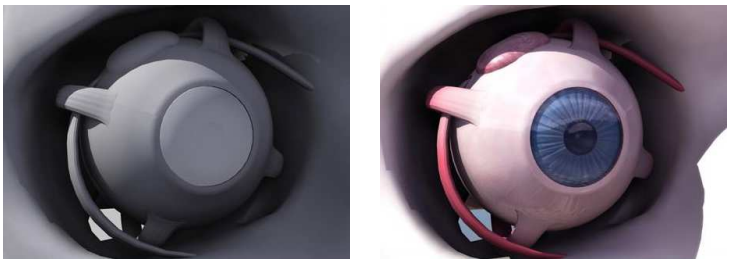
Our current work on lighting design does not take into account variations in color or material properties such as albedo and this should be useful to consider in the future. In addition to lighting design for a single object, automatically designing lighting environments for a scene with multiple objects is also a highly promising area for future research.

5.3 Multi-attribute Saliency

We are currently defining mesh saliency using mean curvature. It should be possible to improve this by using better measures of shape, such as principal curvatures. We can consider the directional differences of geometric features by using principal curvatures.

Our current definition of mesh saliency considers only geometry. However, realistic rendering involves other appearance attributes such as color, texture, and reflectance. In many cases, these properties are more important than the geometry especially when the geometry has little variation across different materials. In Figure 5.1, the iris and pupil are salient when we consider their colors. Generalizing mesh saliency to encompass these appearance attributes should be an important direction for further research. Scale selection is another important open problem.

In Chapter 4, we use larger scales than the operator used in Chapter 3 for considering larger local regions. More general scale selection method is important for consistently providing a suitable saliency for applications.



(a) (b)

Figure 5.1: *(a) Black and white rendering of an eye, and (b) color rendering of an eye. Images are from DE ESPONA 3D Enciclopedia (http://www.deespona.com/).*

5.4 Validation of Saliency

Our method for computing mesh saliency uses a center-surround mechanism which is inspired by the human visual system and applied to measure visual attention in two-dimensional images [41]. We use a center-surround operator since it is a straightforward approach for identifying regions that are unique relative to their surroundings. For two-dimensional images it has been proven that there is a significant correlation between the computed salience and the fixation locations where humans focus their attention [68]. It is plausible that a center-surround operator applied to 3D objects may also capture the regions that humans will find salient. Our experiments on simplification and viewpoint selection provide preliminary indications that this may be true.

It would be useful to validate the correlation between the computed mesh

saliency and the human perceptual attention for 3D objects. One possible way to do this is to use an eye-tracking system to determine the regions on 3D objects that elicit greater visual attention and contrast this with their computed saliency using methods such as ours. Another approach would be to compute 2D saliency maps of the rendered images of 3D objects and compare them to corresponding mesh saliency. Since the final goal of many graphics applications is to generate a rendered image, this can validate the usefulness of mesh saliency specifically for applications such as viewpoint selection and salient lighting, where a rendered image at a given viewing direction is important.

5.5 Task-Specific Saliency

Figure 5.2: *An example of a visual surveillance system identifying a human moving. The image is from Davis* et al. *[16].*

Detecting unusual activities is one of the active research areas in computer vision. This includes detecting suspicious motion in surveillance videos. Recent research in vision has also addressed ignoring periodic motion such as waving trees. This can be used for segmenting the foreground and the background [51]. Applying the core concept of saliency here has much potential for future research. Another interesting issue is the visual presentation of the results of these methods, that are

currently visualized by listing top candidates or explicitly specifying the detected locations as shown in Figure 5.2 [16]. However, these visualization methods might have false negatives. Explicit specifications such as the rectangle in Figure 5.2 are very salient and will capture the viewer's attention. This will make other regions unnoticeable as we can see from Braun [11]'s observation that non-salient regions next to the salient region are not easily noticeable. If we make the saliency of candidates in the image proportional to their confidence levels, the viewer's perception is also likely to be proportional to them. For this, we can enhance the contrast proportional to the confidence levels as in Chapter 4.

5.6 Saliency-based Light Collages

The Light Collages system does not at present incorporate any notion of perceptual saliency in deciding how and where to illuminate a scene. Saliency-based Light Collages is likely to emerge as another area for further research.

Light Collages proceeds in three stages. In the first stage we use a surface-curvature-based watershed algorithm to partition the input mesh into surface patches. In the second stage we define a per-patch light-placement function based on the view direction and average patch curvature. The light placement function models the appropriateness of light directions for illuminating the model. This is done by using the curvature-based segmentation as well as the diffuse and specular illumination at every vertex. In the third stage we aggregate the per-patch light-placement functions and identify a set of suitable light directions. These light directions are then

assigned to patches based on an objective function that minimizes the differences between the diffuse illumination and the local curvature intensity and maximizes the likelihood of having specular highlights on highly curved regions.

We can use mesh saliency to improve each of the three stages – mesh segmentation, light placement, and light assignment. At each stage, we can replace the use of curvature with mesh saliency. For segmentation, we can use mesh saliency for the delimiting values in the watershed algorithm. For light placement and assignment, we can use saliency for computing light placement function and for judging which light has to be assigned to which patch. We expect that the rendered image will show bright and highlighted salient regions and dimmed non-salient regions.

5.7 Saliency-based Segmentation

Mesh segmentation [48] is another mesh processing operation that could benefit from a saliency map that assigns different priorities to different regions of a mesh. Specifically it could benefit the models with repeated patterns such as the Ecoli membrane channel as shown in Figure 4.2. Curvature-based segmentation method would detect fine structures of individual atoms, but might fail to find large structures such as the central channel and the oblique clefts. Much as we find those structures salient in Chapter 4, saliency-based segmentation should be able to segment them.

BIBLIOGRAPHY

[1] A. Agarwala, M. Dontcheva, M. Agrawala, S. Drucker, A. Colburn, B. Curless, D. Salesin, and M. Cohen. Interactive digital photomontage. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2004)*, 23(3):294–302, 2004.

[2] M. Agrawala and F. Durand. Guest editors' introduction: Smart depiction for visual communication. *IEEE Computer Graphics and Applications*, 25(3):20–21, 2005.

[3] D. Akers, F. Losasso, J. Klingner, M. Agrawala, J. Rick, and P. Hanrahan. Conveying shape and features with image-based relighting. In *Proceedings of IEEE Visualization*, pages 349 – 354, 2003.

[4] G. Al-Regib, Y. Altunbasak, and J. Rossignac. Error-resilient transmission of 3D models. *ACM Transactions on Graphics*, 24(2):182–208, 2005.

[5] S. Anderson and M. Levoy. Unwrapping and visualizing cuneiform tablets. *IEEE Computer Graphics and Applications*, 22(6):82–88, 2002.

[6] M. Ashikhmin. A tone mapping algorithm for high contrast images. In *Eurographics Workshop on Rendering*, pages 1–11, 2002.

[7] R. Basri and D. Jacobs. Lambertian reflectance and linear subspaces. In *Proceedings of the Eighth International Conference On Computer Vision*, pages 383–390, July 9–12 2001.

[8] W. Baxter and M. Lin. A versatile interactive 3D brush model. In *Proceedings of Pacific Graphics*, pages 319–328, 2004.

[9] M. A. Blanco, M. Flórez, and M. Bermejo. Evaluation of the rotation matrices in the basis of real spherical harmonics. *Journal of Molecular Structure (Theochem)*, 419:19–27, 1997.

[10] V. Blanz, M. J. Tarr, and H. H. Bülthoff. What object attributes determine canonical views? *Perception*, 28(5):575–599, 1999.

[11] J. Braun. Visual search among items of different salience: removal of visual attention mimics a lesion in extrastriate area V4. *Journal of Neuroscience*, 14(2):554–567, 1994.

[12] P. Cavanagh. Pictorial art and vision. *MIT Encyclopedia of the Cognitive Sciences*, pages 644–646, 1999.

[13] L. Chen, X. Xie, X. Fan, W. Ma, H. Zhang, and H. Zhou. A visual attention model for adapting images on small displays. *ACM Multimedia Systems Journal*, 9(4):353–364, 2003.

[14] C. H. Choi, J. Ivanic, M. S. Gordon, and K. Ruedenberg. Rapid and stable determination of rotation matrices between spherical harmonics by direct recursion. *Journal of Chemical Physics*, 111(19):8825–8831, 1999.

[15] A. C. Costa, A. A. de Sousa, and F. N. Ferreira. Lighting design: A goal based approach using optimization. In *Proceedings of Eurographics Workshop on Rendering Techniques*, pages 317–328, 1999.

[16] L. Davis, S. Fejes, D. Harwood, Y. Yacoob, I. Haritaoglu, and M. Black. Visual surveillance of human activity. In *Proceedings of Asian Conference on Computer Vision*, pages 267–274, 1998.

[17] D. DeCarlo and A. Santella. Stylization and abstraction of photographs. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2002)*, 21(3):769–776, 2002.

[18] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classifiation, 2nd Ed.* John Wiley & Sons, Inc., New York, 2001.

[19] F. Durand and J. Dorsey. Fast bilateral filtering for the display of high dynamic range image. In *Proceedings of ACM SIGGRAPH*, pages 257–265, 2002.

[20] H.-H. Ehricke, K. Donner, W. Koller, and W. Straßer. Visualization of vasculature from volume data. *Computers & Graphics*, 18(3):395–406, 1994.

[21] J. Elder, S. Trithart, G. Pintilie, and D. MacLean. Rapid processing of cast and attached shadows. *Perception*, 33(11):1319 – 1338, 2004.

[22] R. Fattal, D. Lischinski, and M. Werman. Gradient domain high dynamic range compression. In *Proceedings of ACM SIGGRAPH*, 2002.

[23] S. Fleishman, D. Cohen-Or, and D. Lischinski. Automatic camera placement for image-based modeling. In *Proceedings of the 7th Pacific Conference on Computer Graphics and Applications (PG 1999)*, pages 12–20, 1999.

[24] S. Frintrop, A. Nüchter, and H. Surmann. Visual attention for object recognition in spatial 3D data. In *2nd International Workshop on Attention and Performance in Computational Vision (WAPCV 2004)*, pages 75–82, 2004.

[25] M. Garland and P. Heckbert. Surface simplification using quadric error metrics. In *Proceedings of ACM SIGGRAPH*, pages 209–216, 1997.

[26] A. Girshick, V. Interrante, S. Haker, and T. Lemoine. Line direction matters: an argument for the use of principal directions in 3D line drawings. In *Proceedings of the First International Symposium on Non-Photorealistic Animation and Rendering*, pages 43 – 52, 2000.

[27] E. H. Gombrich. *The Heritage of Appelles.* Oxford: Phaidon Press, 1976.

[28] A. Gooch, B. Gooch, P. Shirley, and E. Cohen. A non-photorealistic lighting model for automatic technical illustration. In *Proceedings of ACM SIGGRAPH*, pages 447–452, 1998.

[29] A. A. Gooch, S. C. Olsen, J. Tumblin, and B. Gooch. Color2Gray: Salience-preserving color removal. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2005)*, 24(3), 2005. (to appear).

[30] B. Gooch, E. Reinhard, C. Moulding, and P. Shirley. Artistic composition for image creation. In *Proceedings of Eurographics Workshop on Rendering Techniques*, pages 83–88, 2001.

[31] S. Gumhold. Maximum entropy light source placement. In *Proceedings of IEEE Visualization*, pages 275–282, 2002.

[32] G. Guy and G. Medioni. Inferring global perceptual contours from local features. *International Journal of Computer Vision*, 20(1–2):113–133, 1996.

[33] M. Halle and J. Meng. LightKit: A lighting system for effective visualization. In *Proceedings of IEEE Visualization*, pages 363 – 370, 2003.

[34] J. Hamel. *A New Lighting Model for Computer Generated Line Drawings*. PhD thesis, Otto-von-Guericke-Universität Magdeburg, Germany, 2000.

[35] R. Hawkes, S. Rushton, and M. Smyth. Update rates and fidelity in virtual environments. *Virtual Reality: Research, Applications and Design*, 1(2):46–51, 1995.

[36] P. S. Heckbert and M. Garland. Optimal triangulation and quadric-based surface simplification. *Computational Geometry*, 14:49–65, 1999.

[37] M. Hisada, A. G. Belyaev, and T. L. Kunii. A skeleton-based approach for detection of perceptually salient features on polygonal surfaces. *Computer Graphics Forum*, 21(4):689–700, 2002.

[38] H. Hoppe. Progressive meshes. *Proceedings of ACM SIGGRAPH*, pages 99–108, 1996.

[39] S. Howlett, J. Hamill, and C. O'Sullivan. An experimental approach to predicting saliency for simplified polygonal models. In *Proceedings of the 1st Symposium on Applied Perception in Graphics and Visualization*, pages 57–64, 2004.

[40] L. Itti and C. Koch. Computational modelling of visual attention. *Nature Reviews Neuroscience*, 2(3):194–203, 2001.

[41] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11):1254–1259, 1998.

[42] J. Ivanic and K. Ruedenberg. Rotation matrices for real spherical harmonics. *Journal of Physical Chemistry A*, 102(45):9099–9100, 1998.

[43] B. Kachar. Asymmetric illumination contrast: A method of image formation for video light microscopy. *Science*, 227(4688):766–768, 1985.

[44] J. Kahrs, S. Calahan, D. Carson, and S. Poster. Pixel cinematography: A lighting approach for computer graphics. In *ACM SIGGRAPH Course Notes*, 1996.

[45] R. D. Kalnins, L. Markosian, B. J. Meier, M. A. Kowalski, J. C. Lee, P. L. Davidson, M. Webb, J. F. Hughes, and A. Finkelstein. WYSIWYG NPR: Drawing strokes directly on 3D models. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2002)*, 21(3):755–762, July 2002.

[46] T. Kamada and S. Kawai. A simple method for computing general position in displaying three-dimensional objects. *Computer Vision, Graphics, Image Processing*, 41(1):43–56, 1988.

[47] Z. Karni and C. Gotsman. Spectral compression of mesh geometry. In *Proceedings of ACM SIGGRAPH*, pages 279–286, 2000.

[48] S. Katz and A. Tal. Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2003)*, 22(3), 2003.

[49] J. K. Kawai, J. S. Painter, and M. F. Cohen. Radiooptimization - Goal Based Rendering. In *Proceedings of ACM SIGGRAPH*, pages 147–154, 1993.

[50] Y. Kho and M. Garland. User-guided simplification. In *Symposium on Interactive 3D Graphics*, pages 123–126, 2003.

[51] K. Kim, T. H. Chalidabhongse, D. Harwood, and L. Davis. Real-time foreground-background segmentation using codebook model. *Real-Time Imaging*, 11(3):167–256, 2005.

[52] S.-J. Kim, S.-K. Kim, and C.-H. Kim. Discrete differential error metric for surface simplification. In *Proceedings of 10th Pacific Conference on Computer Graphics and Applications (PG 2002)*, pages 276 – 283, 2002.

[53] L. Kobbelt, S. Campagna, J. Vorsatz, and H.-P. Seidel. Interactive multi-resolution modeling on arbitrary meshes. In *Proceedings of ACM SIGGRAPH*, pages 105–114, 1998.

[54] C. Koch and S. Ullman. Shifts in selective visual attention: towards the underlying neural circuitry. *Human Neurobiology*, 4:219–227, 1985.

[55] C. H. Lee, X. Hao, and A. Varshney. Light collages: Lighting design for effective visualization. In *Proceedings of IEEE Visualization*, pages 281–288, 2004.

[56] C. H. Lee, X. Hao, and A. Varshney. Geometry-dependent lighting. In *IEEE Transactions on Visualization and Computer Graphics*, 2005. (to appear).

[57] C. H. Lee, A. Varshney, and D. Jacobs. Mesh saliency. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2005)*, 24(3), 2005. (to appear).

[58] D. Luebke and B. Hallen. Perceptually driven simplification for interactive rendering. In *Proceedings of Eurographics Workshop on Rendering Techniques*, pages 223 – 234, 2001.

[59] D. Luebke, M. Reddy, J. Cohen, A. Varshney, B. Watson, and R. Huebner. *Level of Detail for 3D Graphics*. Morgan Kaufman, 2003.

[60] A. P. Mangan and R. T. Whitaker. Partitioning 3D surface meshes using watershed segmentation. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):308–321, 1999.

[61] R. Mantiuk, K. Myszkowski, and S. Pattanaik. Attention guided MPEG compression for computer animations. In *Proceedings of the 19th Spring Conference on Computer Graphics*, pages 239–244, 2003.

[62] J. Marks, B. Andalman, P. A. Beardsley, W. Freeman, S. Gibson, J. Hodgins, T. Kang, B. Mirtich, H. Pfister, W. Ruml, K. Ryall, J. Seims, and S. Shieber. Design galleries: a general approach to setting parameters for computer graphics and animation. In *Proceedings of ACM SIGGRAPH*, pages 389–400, 1997.

[63] G. Medioni and G. Guy. Inference of surfaces, curves and junctions from sparse, noisy 3-D data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(11):1265–1277, 1997.

[64] M. Meyer, M. Desbrun, P. Schröder, and A. H. Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and Mathematics III (Proceedings of VisMath 2002)*, pages 35–54, Berlin (Germany), 2003. Springer Verlag.

[65] R. Milanese, H. Wechsler, S. Gil, J. Bost, and T. Pun. Integration of bottom-up and top-down cues for visual attention using non-linear relaxation. In *Proceedings of IEEE Computer Vision and Pattern Recognition*, pages 781–785, 1994.

[66] G. Miller. Efficient algorithms for local and global accessibility shading. In *Proceedings of ACM SIGGRAPH*, pages 319–326, 1994.

[67] Y. Ostrovsky, P. Cavanagh, and P. Sinha. Perceiving illumination inconsistencies in scenes. Technical Report AIM-2001-029, MIT Artificial Intelligence Laboratory, 2001.

[68] D. Parkhurst, K. Law, and E. Niebur. Modeling the role of salience in the allocation of overt visual attention. *Vision Research*, 42(1):107–123, 2002.

[69] S. N. Pattanaik, J. A. Ferwerda, M. D. Fairchild, and D. P. Greenberg. A multiscale model of adaptation and spatial vision for realistic image display. In *Proceedings of ACM SIGGRAPH*, pages 287–298, 1998.

[70] F. Pellacini, P. Tole, and D. P. Greenberg. A user interface for interactive cinematic shadow design. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2002)*, 21(3):563 – 566, 2002.

[71] P. Poulin and A. Fournier. Lights from highlights and shadows. In *Symposium on Interactive 3D Graphics*, pages 31–38, 1992.

[72] P. Poulin, K. Ratib, and M. Jacques. Sketching shadows and highlights to position lights. In *Proceedings of the Conference on Computer Graphics International*, pages 56–64, 1997.

[73] C. Privitera and L. Stark. Focused JPEG encoding based upon automatic preidentified regions of interest. In *Proceedings of SPIE, Human Vision and Electronic Imaging IV*, pages 552–558, 1999.

[74] V. Ramachandran. Perception of shape from shading. *Nature*, 331(6152):163 – 166, 1988.

[75] R. Ramamoorthi and P. Hanrahan. A signal-processing framework for inverse rendering. In *Proceedings of ACM SIGGRAPH*, pages 117–128, 2001.

[76] K. Rasche, R. Geist, and J. Westall. Detail preserving reproduction of color images for monochromats and dichromats. *IEEE Computer Graphics & Applications*, 25(3):22–30, 2005.

[77] R. Raskar, K.-H. Tan, R. Feris, J. Yu, and M. Turk. Non-photorealistic camera: Depth edge detection and stylized rendering using multi-flash imaging.

ACM Transactions on Graphics (Proceedings of SIGGRAPH 2004), 23(3):679–688, 2004.

[78] M. Reddy. Perceptually optimized 3D graphics. *IEEE Computer Graphics and Applications*, 21(5):68–75, 2001.

[79] E. Reinhard, M. Stark, P. Shirley, and J. Ferwerda. Photographic tone reproduction for digital images. In *Proceedings of ACM SIGGRAPH*, 2002.

[80] R. A. Rensink, J. K. O'Regan, and J. J. Clark. To see or not to see: The need for attention to perceive changes in scenes. *Psychological Science*, 8(5):368–373, 1997.

[81] R. Rosenholtz. A simple saliency model predicts a number of motion popout phenomena. *Vision Research*, 39(19):3157–3163, 1999.

[82] C. Schoeneman, J. Dorsey, B. Smits, J. Arvo, and D. Greenberg. Painting with light. In *Proceedings of ACM SIGGRAPH*, pages 143–146, 1993.

[83] R. Shacked and D. Lischinski. Automatic lighting design using a perceptual quality metric. *Computer Graphics Forum (Eurographics 2001)*, 20(3):215–226, 2001.

[84] A. Shashua and S. Ullman. Structural saliency: The detection of globally salient structures using a locally connected network. In *Proceedings of IEEE International Conference on Computer Vision*, pages 321–327, 1988.

[85] P.-P. Sloan, W. Martin, A. Gooch, and B. Gooch. The Lit Sphere: A model for capturing NPR shading from art. In *Proceedings of Graphics Interface*, pages 143–150, 2001.

[86] G. J. F. Smets and K. J. Overbeeke. Trade-off between resolution and interactivity in spatial task performance. *IEEE Computer Graphics and Applications*, 15(5):46–51, 1995.

[87] M. Sousa, F. Samavati, and M. Brunn. Depicting shape features with directional strokes and spotlighting. In *Proceedings of Computer Graphics International*, pages 214 – 221, 2004.

[88] A. James Stewart. Vicinity shading for enhanced perception of volumetric data. In *Proceedings of IEEE Visualization*, pages 355 – 362, 2003.

[89] S. Stoev and W. Straßer. A case study on automatic camera placement and motion for visualizing historical data. In *Proceedings of IEEE Visualization*, pages 545–548, 2002.

[90] T. Strothotte and S. Schlechtweg. *Non-Photorealistic Computer Graphics: Modeling, Rendering, and Animation*. Morgan Kaufmann, San Francisco, 2002.

[91] B. Suh, H. Ling, B. B. Bederson, and D. W. Jacobs. Automatic thumbnail cropping and its effectiveness. *CHI Letters (UIST 2003)*, 5(2):95–104, 2003.

[92] S. I. Sukharev, W. J. Sigurdson, C. Kung, and F. Sachs. Energetic and spatial parameters for gating of the bacterial large conductance mechanosensitive channel, MscL. *Journal of General Physiology*, 113(4):525–540, 1999.

[93] G. Taubin. Estimating the tensor of curvature of a surface from a polyhedral approximation. In *Proceedings of IEEE International Conference on Computer Vision*, pages 902–907, 1995.

[94] J. K. Tsotsos, S. M. Culhane, W. Y. K. Wai, Y. H. Lai, N. Davis, and F. Nuflo. Modeling visual-attention via selective tuning. *Artificial Intelligence*, 78(1-2):507–545, 1995.

[95] J. Tumblin and G. Turk. Lcis: A boundary hierarchy for detail-preserving contrast reduction. In *Proceedings of ACM SIGGRAPH*, pages 83–90, 1999.

[96] G. Turk. Re-tiling polygon surfaces. In *Proceedings of ACM SIGGRAPH*, pages 55–64, 1992.

[97] P.-P. Vázquez, M. Feixas, M. Sbert, and A. Llobet. Viewpoint entropy: a new tool for obtaining good views of molecules. In *Proceedings of the Symposium on Data Visualisation (VISSYM 2002)*, pages 183–188, 2002.

[98] K. Watanabe and A. G. Belyaev. Detection of salient curvature features on polygonal surfaces. *Computer Graphics Forum (Eurographics 2001)*, 20(3):385–392, 2001.

[99] B. Watson, N. Walker, and L. F. Hodges. Supra-threshold control of peripheral LOD. *ACM Transactions on Graphics*, 23(3):750–759, 2004.

[100] D. Weinshall and M. Werman. On view likelihood and stability. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(2):97–108, 1997.

[101] H. Yee, S. Pattanaik, and D. P. Greenberg. Spatiotemporal sensitivity and visual attention for efficient rendering of dynamic environments. *ACM Transactions on Graphics*, 20(1):39–65, 2001.

[102] Y. Yu, P. Debevec, J. Malik, and T. Hawkins. Inverse global illumination: recovering reflectance models of real scenes from photographs. In *Proceedings of ACM SIGGARPH*, pages 215–224, 1999.